Reasoning in the OWL 2 Full Ontology Language using First-Order Automated Theorem Proving

Michael Schneider[1*] and Geoff Sutcliffe[2]

[1] FZI Research Center for Information Technology, Germany
[2] University of Miami, USA

arXiv:1108.0155v1 [cs.AI] 31 Jul 2011

Abstract. OWL 2 has been standardized by the World Wide Web Consortium (W3C) as a family of ontology languages for the Semantic Web. The most expressive of these languages is OWL 2 Full, but to date no reasoner has been implemented for this language. Consistency and entailment checking are known to be undecidable for OWL 2 Full. We have translated a large fragment of the OWL 2 Full semantics into first-order logic, and used automated theorem proving systems to do reasoning based on this theory. The results are promising, and indicate that this approach can be applied in practice for effective OWL reasoning, beyond the capabilities of current Semantic Web reasoners.
This is an *extended version* of a paper with the same title that has been published at CADE 2011, LNAI 6803, pp. 446–460. The extended version provides appendices with additional resources that were used in the reported evaluation.

Key words: Semantic Web, OWL, First-order logic, ATP

1 Introduction

The Web Ontology Language OWL 2 [16] has been standardized by the World Wide Web Consortium (W3C) as a family of ontology languages for the Semantic Web. OWL 2 includes OWL 2 DL [10], the OWL 2 RL/RDF rules [9], as well as OWL 2 Full [12]. The focus of this work is on reasoning in OWL 2 Full, the most expressive of these languages. So far, OWL 2 Full has largely been ignored by the research community, and no reasoner has been implemented for this language.

OWL 2 Full does not enforce any of the numerous syntactic restrictions of the description logic-style language OWL 2 DL. Rather, OWL 2 Full treats arbitrary RDF graphs [7] as valid input ontologies, and can safely be used with weakly structured RDF data as is typically found on the Web. Further, OWL 2 Full provides for reasoning outside the scope of OWL 2 DL and the OWL 2 RL/RDF rules, including sophisticated reasoning based on meta-modeling. In addition, OWL 2 Full is semantically fully compatible with RDFS [5] and also with the OWL 2 RL/RDF rules, and there is even a strong semantic correspondence [12]

* Partially supported by the projects *SEALS* (European Commission, EU-IST-2009-238975) and *THESEUS* (German Federal Ministry of Economics and Technology, FK OIMQ07019).

with OWL 2 DL, roughly stating that any OWL 2 DL conclusion can be reflected in OWL 2 Full. This makes OWL 2 Full largely interoperable with the other OWL 2 languages, and allows an OWL 2 Full reasoner to be combined with most existing OWL reasoners to provide higher syntactic flexibility and semantic expressivity in reasoning-enabled applications.

Due to its combination of flexibility and expressivity, OWL 2 Full is computationally undecidable with regard to consistency and entailment checking [8]. While there cannot be any complete decision procedure for OWL 2 Full, the question remains to what extent practical OWL 2 Full reasoning is possible. This paper presents the results of a series of experiments about reasoning in OWL 2 Full using first-order logic (FOL) theorem proving. A large fragment of the OWL 2 Full semantics has been translated into a FOL theory, and automated theorem proving (ATP) systems have been used to do reasoning based on this theory. The primary focus of these experiments was on the question of what can be achieved at all; a future study may shift the focus to efficiency aspects.

The basic idea used in this work is not new. An early application of this approach to a preliminary version of RDF and a precursor of OWL was reported by Fikes et al. [2]. That work focused on identifying technical problems in the original language specifications, rather than on practical reasoning. Hayes [4] provided fairly complete translations of RDF(S) and OWL 1 Full into Common Logic, but did not report on any reasoning experiments. This gap was filled by Hawke's reasoner *Surnia* [3], which applied an ATP system to an FOL axiomatisation of OWL 1 Full. For unknown reasons, however, Surnia performed rather poorly on reasoning tests [17]. Comparable studies have been carried out for ATP-based OWL DL reasoning, as for *Hoolet* [15], an OWL DL reasoner implemented on top of a previous version of the Vampire ATP system (`http://www.vprover.org`). The work of Horrocks and Voronkov [6] addresses reasoning over large ontologies, which is crucial for practical Semantic Web reasoning. Finally, [1] reports on some historic knowledge representation systems using ATP for description logic-style reasoning, such as *Krypton* in the 1980s.

All these previous efforts are outdated, in that they refer to precursors of OWL 2 Full, and appear to have been discontinued after publication. The work reported in this paper refers to the current specification of OWL 2 Full, and makes a more extensive experimental evaluation of the FOL-based approach than any previous work. Several aspects of OWL 2 Full reasoning have been studied: the degree of language coverage of OWL 2 Full; semantic conclusions that are characteristic specifically of OWL 2 Full; reasoning on large data sets; and the ability of first-order systems to detect non-entailments and consistent ontologies in OWL 2 Full. The FOL-based results have been compared with the results of a selection of well-known Semantic Web reasoners, to determine whether the FOL-based approach is able to add significant value to the state-of-the-art in Semantic Web reasoning.

This paper is organized as follows: Section 2 provides an introduction to the technologies used in this paper. Section 3 describes the FOL-based reasoning approach. Section 4 describes the evaluation setting, including the test data,

the reasoners, and the computers used in the experiments. The main part of the paper is Section 5, which presents the results of the experiments. Section 6 concludes, and gives an outlook on possible future work. The appendices present the raw result data underlying the evaluation results (A); the complete test suite of "characteristic OWL 2 Full conclusions" that has been used in the evaluation (B); and an example showing how RDF data and the semantics of OWL 2 Full have been translated into the first-order logic formalism (C).

2 Preliminaries

2.1 RDF and OWL 2 Full

OWL 2 Full is specified as the language that uses the OWL 2 RDF-Based Semantics [12] to interpret arbitrary RDF graphs. RDF graphs are defined by the RDF Abstract Syntax [7]. The OWL 2 RDF-Based Semantics is defined as a semantic extension of the RDF Semantics [5].

According to the RDF Abstract Syntax, an *RDF graph* G is a set of RDF triples: $G = \{t_1, \ldots, t_n\}$. Each *RDF triple* t is given as an ordered ternary tuple $t = s\,p\,o$ of *RDF nodes*. The RDF nodes s, p, and o are called the *subject*, *predicate*, and *object* of the triple t, respectively. Each RDF node is either a *URI*, a (plain, language-tagged or typed) *literal*, or a *blank node*.

The *RDF Semantics* is defined on top of the RDF Abstract Syntax as a model theory for arbitrary RDF graphs. For an *interpretation* I and a *domain* U, a URI denotes an individual in the domain, a literal denotes a concrete data value (also considered a domain element), and a blank node is used as an existentially quantified variable indicating the existence of some domain element. The meaning of a triple $t = s\,p\,o$ is a truth value of the relationship $\langle I(s), I(o)\rangle \in \mathrm{IEXT}(I(p))$, where IEXT is a mapping from domain elements that are *properties* to associated binary relations. The meaning of a graph $G = \{t_1, \ldots, t_n\}$ is a truth value determined by the conjunction of the meaning of all the triples, taking into account the existential semantics of blank nodes occurring in G. If an RDF graph G is true under an interpretation I, then I *satisfies* G. An RDF graph G is *consistent* if there is an interpretation I that satisfies G. An RDF graph G *entails* another RDF graph H if every interpretation I that satisfies G also satisfies H.

Whether an interpretation satisfies a given graph is primarily determined by a collection of model-theoretic *semantic conditions* that constrain the mapping IEXT. There are different sets of model-theoretic semantic conditions for the different semantics defined by the RDF Semantics specification. For example, the semantics of class subsumption in *RDFS* is defined mainly by the semantic condition defined for the RDFS vocabulary term `rdfs:subClassOf`:

$$\langle c, d\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf})) \Rightarrow c, d \in IC \wedge \mathrm{ICEXT}(c) \subseteq \mathrm{ICEXT}(d)$$

where "c" and "d" are universally quantified variables. Analogous to the mapping IEXT, the mapping ICEXT associates *classes* with subsets of the domain. The two mappings are responsible for the *metamodeling capabilities* of RDFS

and its semantic extensions: Although the quantifiers in the RDFS semantic conditions range over exclusively domain elements, which keeps RDFS in the realm of first-order logic, the associations provided by the two mappings allow domain elements (properties and classes) to indirectly refer to sets and binary relations. This enables a limited but useful form of higher order-style modeling and reasoning.

The *OWL 2 RDF-Based Semantics*, i.e. the semantics of OWL 2 Full, extends the RDF Semantics specification by additional semantic conditions for the OWL-specific vocabulary terms, such as `owl:unionOf` and `owl:disjointWith`.

2.2 FOL, the TPTP language, and ATP

The translation of the OWL 2 Full semantics is to classical untyped first-order logic. The concrete syntax is the TPTP language [14], which is the de facto standard for state-of-the-art ATP systems for first-order logic. The ATP systems used in the evaluation were taken from their web sites (see Section 4.3) or from the archives of the 5th IJCAR ATP System Competition, CASC-J5 (`http://www.tptp.org/CASC/J5/`). Most of the systems are also available online as part of the SystemOnTPTP service (`http://www.tptp.org/cgi-bin/SystemOnTPTP/`).

3 Approach

Each of the model-theoretic semantic conditions of the OWL 2 Full semantics is translated into a corresponding FOL axiom. The result is an axiomatization of OWL 2 Full. The RDF graphs to reason about are also converted into FOL formulae. In the case of *consistency checking* there is a single RDF graph that is converted into a FOL axiom, for which satisfiability needs to be checked. In the case of *entailment checking*, there is a premise graph that is converted into a FOL axiom, and a conclusion graph that is converted into a FOL conjecture. The FOL formulae (those representing the input RDF graphs and those building the FOL axiomatization of the OWL 2 Full semantics) are passed to an ATP system, which tries to prove the conclusion or establish consistency.

We apply a straight-forward *translation of the semantic conditions*, making use of the fact that all semantic conditions have the form of FOL formulae. A semantic relationship of the form "$\langle s, o\rangle \in \mathrm{IEXT}(p)$" that appears within a semantic condition is converted into an atomic FOL formula of the form "$\mathrm{iext}(p, s, o)$". Likewise, a relationship "$x \in \mathrm{ICEXT}(c)$" is converted into "$\mathrm{icext}(c, x)$". Apart from this, the basic logical structure of the semantic conditions is retained. For example, the semantic condition specifying RDFS class subsumption shown in Section 2.1 is translated into

$$\begin{aligned}\forall c, d : [\ & \mathrm{iext}(\texttt{rdfs:subClassOf}, c, d) \Rightarrow \\ & (\ \mathrm{ic}(c) \wedge \mathrm{ic}(d) \wedge \forall x : (\mathrm{icext}(c, x) \Rightarrow \mathrm{icext}(d, x))\ )\ ]\end{aligned}$$

The *translation of RDF graphs* amounts to converting the set of triples "$s\,p\,o$" into a conjunction of corresponding "$\mathrm{iext}(p, s, o)$" atoms. A *URI* occurring in an

RDF graph is converted into a constant. An *RDF literal* is converted into a function term, with a constant for the literal's lexical form as one of its arguments. Different functions are used for the different kinds of literals: function terms for *plain literals* have arity 1; function terms for *language-tagged literals* have a constant representing the language tag as their second argument; function terms for *typed literals* have a constant for the datatype URI as their second argument. For each *blank node*, an existentially quantified variable is introduced, and the scope of the corresponding existential quantifier is the whole conjunction of the "iext" atoms. For example, the RDF graph

```
_:x rdf:type foaf:Person .
_:x foaf:name "Alice"^^xsd:string .
```

which contains the blank node "`_:x`", the typed literal "`"Alice"^^xsd:string`", and the URIs "`rdf:type`", "`foaf:Person`", and "`foaf:name`", is translated into the FOL formula

$$\begin{aligned}\exists x : [\ &\mathrm{iext}(\mathtt{rdf{:}type}, x, \mathtt{foaf{:}Person}) \wedge \\ &\mathrm{iext}(\mathtt{foaf{:}name}, x, \mathtt{literal}_{\mathtt{typed}}(\mathtt{Alice}, \mathtt{xsd{:}string}))\ ]\end{aligned}$$

4 Evaluation Setting

This section describes the evaluation setting: the OWL 2 Full axiomatization, the test cases, the reasoners, and the computing resources. *Supplementary material* including the axiomatizations, test data, raw results, and the software used for this paper can be found online at:
`http://www.fzi.de/downloads/ipe/schneid/cade2011-schneidsut-owlfullatp.zip`.

4.1 The FOL Axiomatization and RDF Graph Conversion

Following the approach described in Section 3, most of the normative semantic conditions of the OWL 2 Full semantics have been converted into the corresponding FOL axioms, using the TPTP language [14]. The main omission is that most of the semantics concerning *reasoning on datatypes* has not been treated, as we were only interested in evaluating the "logical core" of the language. All other language features of OWL 2 Full were covered in their full form, with a restriction that was sufficient for our tests: while OWL 2 Full has many size-parameterized language features, for example the intersection of arbitrarily many classes, our axiomatization generally supports these language feature schemes only up to a size of 3. The resulting FOL axiomatization consists of 558 formulae. The axiom set is fully first-order with equality, but equality accounts for less than 10% of the atoms. The first-order ATP systems used (see Section 4.3) convert the formulae to clause normal form. The resultant clause set is non-Horn. Almost all the clauses are range-restricted, which can result in reasoning that produces mostly ground clauses.

In addition, a converter from RDF graphs to FOL formulae was implemented. This allowed the use of RDF-encoded OWL test data in the experiments, without time consuming and error prone manual conversion.

4.2 Test Data

Two complementary test suites were used for the experiments: one test suite to evaluate the degree of language coverage of OWL 2 Full, and another suite consisting of characteristic conclusions for OWL 2 Full reasoning. For scalability experiments a large set of RDF data was also used.

The Language Coverage Test Suite. For the language coverage experiments, the test suite described in [13] was used.[3] The test suite was created specifically as a conformance test suite for OWL 2 Full and its main sub languages, including RDFS and the OWL 2 RL/RDF rules. The test suite consists of one or more test cases for each of the semantic conditions of the OWL 2 RDF-Based Semantics, i.e., the test suite provides a systematic coverage of OWL 2 Full at a specification level. Most of the test cases are positive entailment and inconsistency tests, but there are also a few negative entailment tests and positive consistency tests. The complete test suite consists of 736 test cases. A large fraction of the test suite deals with datatype reasoning. As the FOL axiomatization has almost no support for datatype reasoning, only the test cases that cover the "logical core" of OWL 2 Full were used. Further, only the positive entailment and inconsistency tests were used. The resultant test suite has 411 test cases.

OWL 2 Full-characteristic Test Cases. In order to investigate the extent of the reasoning possible using the FOL axiomatization, a set of test cases that are characteristic conclusions of OWL 2 Full was created. "Characteristic" means that the test cases represent OWL 2 Full reasoning that cannot normally be expected from either OWL 2 DL reasoning or from reasoners implementing the OWL 2 RL/RDF rules. The test suite consists of 32 tests, with 28 entailment tests and 4 inconsistency tests. There are test cases probing semantic consequences from meta-modeling, annotation properties, the unrestricted use of complex properties, and consequences from the use of OWL vocabulary terms as regular entities (sometimes called "syntax reflection").

Bulk RDF Data. For the scalability experiments, a program that generates RDF graphs of arbitrary size ("bulk RDF data") was written. The data consist of RDF triples using URIs that do not conflict with the URIs in the test cases. Further, no OWL vocabulary terms are used in the data sets. This ensures that adding this bulk RDF data to test cases does not affect the reasoning results.

4.3 Reasoners

This section lists the different reasoning systems that were used in the experiments. The idea behind the selection was to have a small number of represen-

[3] There is an official W3C test suite for OWL 2 at `http://owl.semanticweb.org/page/OWL_2_Test_Cases` (2011-02-09). However, it does not cover OWL 2 Full sufficiently well, and was not designed in a systematic way that allows easy determination of which parts of the language specification are not supported by a reasoner.

tative systems for (i) first-order proving, (ii) first-order model finding, and (iii) OWL reasoning. Details of the ATP systems can be found on their web sites, and (for most) in the system descriptions on the CASC-J5 web site. The OWL reasoners were tested to provide comparisons with existing state of the art Semantic Web reasoners. Unless explicitly stated otherwise, the systems were used in their default modes.

Systems for first-order theorem proving

- ***Vampire 0.6*** (`http://www.vprover.org`). A powerful superposition-based ATP system, including strategy scheduling.
- ***Vampire-SInE 0.6*** A variant of Vampire that always runs the SInE strategy (`http://www.cs.man.ac.uk/~hoderk/sine/desc/`) to select axioms that are expected to be relevant.
- ***iProver-SInE 0.8*** (`http://www.cs.man.ac.uk/~korovink/iprover`). An instantiation-based ATP system, using the SInE strategy, and including strategy scheduling.

Systems for first-order model finding

- ***Paradox 4.0*** (`http://www.cse.chalmers.se/~koen/code/`). A finite model finder, based on conversion to propositional form and the use of a SAT solver.
- ***DarwinFM 1.4.5*** (`http://goedel.cs.uiowa.edu/Darwin`). A finite model finder, based on conversion to function-free first-order logic and the use of the Darwin ATP system.

Systems for OWL reasoning

- ***Pellet 2.2.2*** (`http://clarkparsia.com/pellet`). An OWL 2 DL reasoner that implements a tableaux-based decision procedure.
- ***HermiT 1.3.2*** (`http://hermit-reasoner.com`). An OWL 2 DL reasoner that implements a tableaux-based decision procedure.
- ***FaCT++ 1.5.0*** (`http://owl.man.ac.uk/factplusplus`). An OWL 2 DL reasoner that implements a tableaux-based decision procedure.
- ***BigOWLIM 3.4*** (`http://www.ontotext.com/owlim`). An RDF entailment-rule reasoner that comes with predefined rule sets. The OWL 2 RL/RDF rule set (`owl2-rl`) was used. The commercial "BigOWLIM" variant of the reasoning engine was applied, because it provides inconsistency checking.
- ***Jena 2.6.4*** (`http://jena.sourceforge.net`). A Java-based RDF framework that supports RDF entailment-rule reasoning and comes with predefined rule sets. The most expressive rule set, `OWL_MEM_RULE_INF`, was used.
- ***Parliament 2.6.9*** (`http://parliament.semwebcentral.org`). An RDF triple store with some limited OWL reasoning capabilities. Parliament cannot detect inconsistencies in ontologies.

4.4 Evaluation Environment

Testing was done on computers with a 2.8GHz Intel Pentium 4 CPU, 2GB memory, running Linux FC8. A 300s CPU time limit was imposed on each run.

5 Evaluation Results

This section presents the results of the following reasoning experiments: a *language coverage analysis*, to determine the degree of conformance to the language specification of OWL 2 Full; *"characteristic" OWL 2 Full reasoning* experiments to determine the extent to which distinguishing OWL 2 Full reasoning is possible; some basic *scalability testing*; and several *model finding experiments* to determine whether first-order model finders can be used in practice for the recognition of non-entailments and consistent ontologies. The following markers are used in the result tables to indicate the outcomes of the experiments:

- ***success* ('+'):** a test run that provided the correct result.
- ***wrong* ('–'):** a test run that provided a wrong result, e.g., when a reasoner claims that an entailment test case is a non-entailment.
- ***unknown* ('?'):** a test run that did not provide a result, e.g., due to a processing error or time out.

This section also presents comparative evaluation results for the OWL reasoners listed in Section 4.3. This illustrates the degree to which OWL 2 Full reasoning can already be achieved with existing OWL reasoners, and the added value of our reasoning approach compared to existing Semantic Web technology. This means, for example, that an OWL 2 DL reasoner will produce a wrong result if it classifies an OWL 2 Full entailment test case as a non-entailment. However, this negative evaluation result refers to only the level of conformance with respect to OWL 2 Full reasoning, i.e., the reasoner may still be a compliant implementation of OWL 2 DL.

5.1 Language Coverage

This experiment used the FOL axiomatization with the 411 test cases in the language coverage suite described in Section 4.2. The results of the experiment are shown in Table 1. iProver-SInE succeeded on 93% of the test cases, and Vampire succeeded on 85%. It needs to be mentioned that the results were not perfectly stable. Over several runs the number of successes varied for iProver-SInE between 382 and 386. This is caused by small variations in the timing of strategy changes within iProver-SInE's strategy scheduling.

Figure 1 shows the runtime behavior of the two systems, with the times for successes sorted into increasing order. Both systems take less than 1s for the majority of their successes. Although Vampire succeeded on less cases than iProver-SInE, it is typically faster in the case of a success.

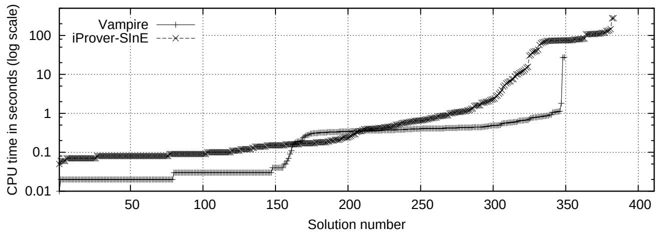

Fig. 1. Language coverage: ordered system times of ATPs.

An analysis of the 28 test cases for which both Vampire and iProver-SInE did not succeed revealed that 14 of them require support for OWL 2 Full language features not covered by the FOL axiomatization, including certain forms of datatype reasoning and support for the RDF container vocabulary [5]. A future version of the axiomatization will encode these parts of the OWL 2 Full semantics, which might lead to improved results. For each of the remaining 14 test cases, subsets of axioms sufficient for a solution were hand-selected from the FOL axiomatization. These axiom sets were generally very small, with up to 16 axioms, and in most cases less than 10 axioms. iProver-SInE succeeded on 13 of these 14 test cases. The remaining test case is a considerably complex one, involving the semantics of qualified cardinality restrictions. It was solved by Vampire. Thus, all test cases were solved except for the 14 that are beyond the current axiomatization.

For comparison, the OWL reasoners listed in Section 4.3 were also tested. The results are shown in Table 2. The OWL 2 DL reasoners Pellet and HermiT both succeeded on about 60% of the test cases. A comparison of the individual results showed that the two reasoners succeeded mostly on the same test cases. Interestingly, although most of the test cases are formally invalid OWL 2 DL ontologies, reasoning rarely resulted in a processing error. Rather, in ca. 40% of the cases, the reasoners wrongly reported a test case to be a non-entailment or a consistent ontology. The third OWL 2 DL reasoner, FaCT++, signaled a processing error more often, and succeeded on less than 50% of the test cases.

The OWL 2 RL/RDF rule reasoner BigOWLIM succeeded on roughly 70% of the test cases. Although the number of successful tests was larger than for

Reasoner	**Success**	**Wrong**	**Unknown**
Vampire	349	0	62
iProver-SInE	383	0	28

Table 1. Language coverage: ATPs with OWL 2 Full axiom set.

Reasoner	**Success**	**Wrong**	**Unknown**
Pellet	237	168	6
HermiT	246	157	8
FaCT++	190	45	176
BigOWLIM	282	129	0
Jena	129	282	0
Parliament	14	373	24

Table 2. Language coverage: OWL reasoners.

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PE	+	+	+	−	−	−	−	−	+	+	−	−	−	−	+	−	−	−	−	+	+	−	−	−	−	+	−	−	?	−	−	−
HE	+	?	+	−	−	?	−	+	+	+	−	−	−	−	+	−	−	−	−	+	+	−	−	+	?	+	−	−	?	−	−	−
FA	+	?	?	?	?	?	?	−	?	+	−	−	−	?	+	?	−	−	−	+	+	?	?	?	?	+	−	?	?	−	−	?
BO	+	−	−	+	−	−	+	+	−	−	+	+	−	−	+	−	−	+	+	−	−	−	−	−	−	−	−	−	−	−	−	−
JE	+	−	−	−	−	+	+	+	−	−	+	−	−	−	−	−	−	+	−	−	−	−	+	−	−	+	−	−	−	−	−	+
PA	+	−	−	−	−	−	−	+	−	−	?	−	−	−	−	−	−	−	?	−	−	−	−	−	−	−	−	−	−	?	?	−

Table 3. Characteristic conclusions: OWL reasoners. PE=Pellet, HE=HermiT, FA=FaCT++, BO=BigOWLIM, JE=Jena, PA=Parliament.

all the OWL 2 DL reasoners, there was a considerable number of test cases for which the OWL 2 DL reasoners were successful but not BigOWLIM, and vice versa. The Jena OWL reasoner, which is an RDF entailment rule reasoner like BigOWLIM, succeeded on about only 30% of the test cases, which is largely due to missing support for OWL 2 features. Finally, Parliament succeeded on only 14 of the test cases. In particular, it did not solve any of the inconsistency test cases. The low success rate reflects the style of "light-weight reasoning" used in many reasoning-enabled RDF triple stores.

5.2 Characteristic OWL 2 Full Conclusions

The test suite of characteristic OWL 2 Full conclusions focuses on semantic consequences that are typically beyond the scope of OWL 2 DL or RDF rule reasoners. This is reflected in Table 3, which presents the results for the OWL reasoning systems. The column numbers correspond to the test case numbers in the test suite. In general, the OWL reasoners show significantly weaker performance on this test suite than on the language coverage test suite. Note that the successful test cases for the OWL 2 DL reasoners (Pellet, HermiT and FaCT++) have only little overlap with the successful test cases for the RDF rule reasoners (BigOWLIM and Jena). Parliament succeeded on only two test cases.

The first two rows of Table 4 show that the ATP systems achieved much better results than the OWL reasoners, using the complete OWL 2 Full axiomatization. iProver-SInE succeeded on 28 of the 32 test cases, and Vampire succeeded on 23. As was done for the language coverage test cases, small subsets of axioms sufficient for each of the test cases were hand-selected from the

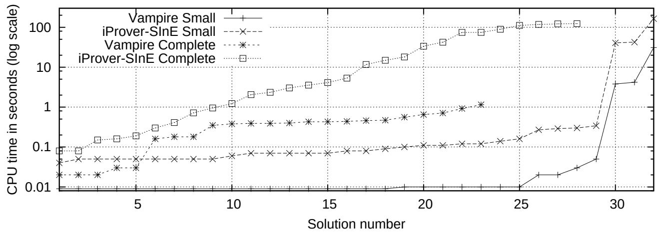

Fig. 2. Characteristic conclusions: ordered system times of ATPs.

FOL axiomatization. As the last two rows of Table 4 show, both ATP systems succeeded on all these simpler test cases.

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
VA/c	+	+	+	+	+	+	+	+	+	?	+	?	?	+	+	+	+	+	+	?	?	?	+	+	?	+	?	?	+	+	+	+
IS/c	+	+	+	+	+	+	+	+	+	+	+	?	?	+	+	+	+	+	+	?	?	+	+	+	+	+	+	+	+	+	+	+
VA/s	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
IS/s	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Table 4. Characteristic conclusions: ATPs with complete and small axiom sets. VA/c=Vampire/complete, IS/c=iProver-SInE/complete, VA/s=Vampire/small, IS/s=iProver-SInE/small.

Figure 2 shows the runtime behavior of the two systems. For the complete axiomatization, Vampire either succeeds in less than 1s or does not succeed. In contrast, iProver's performance degrades more gracefully. The reasoning times using the small-sufficient axiom sets are generally up to several magnitudes lower than for the complete axiomatization. In the majority of cases they are below 1s.

5.3 Scalability

The Semantic Web consists of huge data masses, but single reasoning results presumably often depend on only a small fraction of that data. As a basic test of the ATP systems' abilities to ignore irrelevant background axioms, a set of one million "bulk RDF axioms" (as described in Section 4.2) was added to the test cases of characteristic OWL 2 Full conclusions. This was done using the complete FOL axiomatization, and also the small-sufficient sets of axioms for each test case.

Table 5 shows the results. The default version of Vampire produced very poor results, as is shown in the first and fourth rows of the table. (Strangely, Vampire had two more successes with the complete axiomatization than with the small-sufficient axiom sets. That can be attributed to differences in the strategies selected for the different axiomatizations.) In contrast, as shown in the second,

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
VA/c	+	+	+	?	?	?	?	?	?	?	?	?	?	?	+	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
VS/c	+	+	+	+	+	+	?	+	?	?	+	?	?	?	+	+	?	+	+	?	?	?	+	?	?	+	?	?	?	+	?	+
IS/c	+	+	+	+	+	+	+	+	+	+	+	?	?	+	+	+	+	+	+	?	?	+	+	+	+	+	+	+	+	+	+	+
VA/s	+	?	+	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?	?
VS/s	+	+	+	+	+	+	?	+	+	+	+	+	?	+	+	+	?	+	+	?	?	+	+	+	+	+	+	?	+	+	?	+
IS/s	+	+	+	+	+	+	+	+	+	+	+	+	?	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Table 5. Scalability: ATPs with complete and small axiom sets, 1M RDF triples. VA/c=Vampire/complete, VS/c=Vampire-SInE/complete, IS/c=iProver-SInE/complete, VA/s=Vampire/small, VS/s=Vampire-SInE/small, IS/s=iProver-SInE/small.

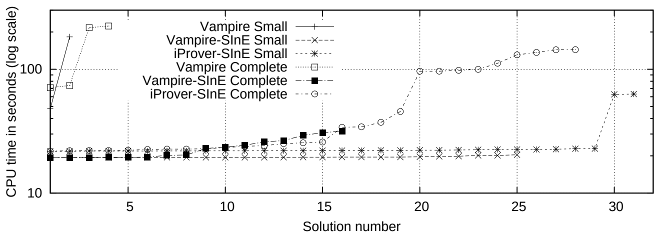

Fig. 3. Scalability: ordered system times of ATPs, 1M RDF triples.

third, fifth and sixth rows, the version of Vampire-SInE and iProver-SInE did much better. The use of the SInE strategy for selecting relevant axioms clearly helps.

Figure 3 shows the runtime behavior of the systems. The bulk axioms evidently add a constant overhead of about 20s to all successes, which is believed to be taken parsing the large files. In an application setting this might be done only once at the start, so that the time would be amortized over multiple reasoning tasks. The step in iProver's performance at the 20th problem is an artifact of strategy scheduling.

The bulk axioms were designed to have no connection to the FOL axiomatization or the RDF graphs. As such, simple analysis of inference chains from the conjecture [11] would be sufficient to determine that the bulk axioms could not be used in a solution. This simplistic approach is methodologically an appropriate way to start testing robustness against irrelevant axioms, and potentially not too far off the reality of Semantic Web reasoning. However, future work using axioms that are not so obviously redundant would properly exercise the power of the SInE approach to axiom selection.

5.4 Model Finding

This section presents the results from experiments concerning the detection of non-entailments and consistent ontologies w.r.t. OWL 2 Full and two of its sub languages: ALCO Full [8] and RDFS [5]. ALCO Full is interesting because it is

	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32
PA/A				+	+	+	+	+			+		?		+	+	+	+	+	?	?	?	?	+	?	?	?	+		+	?	+
PA/R				+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+
DF/R				+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+	+

Table 6. Model Finding: ATPs with ALCO Full and RDFS axiom sets. The black entries indicate positive entailments or inconsistent ontologies. PA/A=Paradox/ALCO Full, PA/R=Paradox/RDFS, DF/R=DarwinFM/RDFS.

a small fragment of OWL 2 Full that is known to be undecidable [8]. RDFS is interesting because it is a minimally meaningful language that shares the main characteristics of OWL 2 Full. The RDFS axioms included the "extensional" semantic extension, as non-normatively defined in Section 4.2 of [5]. Similarly, the original definition of ALCO Full was extended to include extensional RDFS. No report is given for the OWL reasoners, as only the OWL 2 DL reasoners have model-finding capabilities, and not for any of the three languages considered here.

Consistency checking for an RDF graph G w.r.t. some ontology language L corresponds to consistency checking for the combination of a complete axiomatization of L and the FOL translation of G. Hence, a minimum requirement is to confirm that the FOL axiomatization of OWL 2 Full is consistent. Unfortunately, for the OWL 2 Full axiomatization no model finder was able to confirm consistency.[4]

For the ALCO Full axioms, Paradox found a finite model of size 5 in ca. 5s CPU time, while DarwinFM timed out. Paradox was then used on the characteristic OWL 2 Full test cases, with the OWL 2 Full axiomatization replaced by the ALCO Full axioms. As ALCO Full is a sub language of OWL 2 Full, 24 of the 32 test cases are either non-entailments or consistent ontologies, out of which 15 were correctly recognized by Paradox. iProver-SInE was used to confirm that the remaining 8 test cases are positive entailments or inconsistent ontologies. The results are shown in the first row of Table 6.

For the RDFS axioms, analogous experiments were done. Paradox found a finite model of the axioms, of size 1, in about 1s. The consistency was confirmed by DarwinFM in less than 1s. With the OWL 2 Full axiomatization replaced by the RDFS axioms, 29 of the 32 characteristic test cases are non-entailments or consistent ontologies. Paradox and Darwin confirmed all of these, mostly in ca. 1s, with a maximum time of ca. 2s. iProver-SInE confirmed that the remaining 3 test cases are positive entailments or inconsistent ontologies. These results are shown in the second and third rows of Table 6.

[4] This raised the question of whether our positive entailment reasoning results were perhaps due to an inconsistent axiomatization. However, none of the theorem provers was able to establish inconsistency. In addition, the model finders confirmed the consistency of all the small-sufficient axiom sets mentioned in Section 5.2. Hence, it is at least ensured that those positive reasoning results are achievable from consistent subsets of the OWL 2 Full axiomatization.

An interesting observation made during the model finding experiments was that *finite* model finders were effective, e.g., the results of Paradox and DarwinFM above. In contrast, other model finders such as iProver-SAT (a variant of iProver tuned for model finding) and Darwin (the plain Model Evolution core of DarwinFM) were less effective, e.g., taking 80s and 37s respectively to confirm the satisfiability of the RDFS axiom set.

6 Conclusions and Future Work

This paper has described how first order ATP systems can be used for reasoning in the OWL 2 Full ontology language, using a straight-forward translation of the underlying model theory into a FOL axiomatization. The results were obtained from two complementary test suites, one for language coverage analysis and one for probing characteristic conclusions of OWL 2 Full. The results indicate that this approach can be applied in practice for effective OWL reasoning, and offers a viable alternative to current Semantic Web reasoners. Some scalability testing was done by adding large sets of semantically unrelated RDF data to the test case data. While the ATP systems that include the SInE strategy effectively ignored this redundant data, it was surprising that other ATP systems did not use simple reachability analysis to detect and ignore this bulk data – this suggests an easy way for developers to adapt their systems to such problems.

In contrast to the successes of the ATP systems proving theorems, model finders were less successful in identifying non-entailments and consistent ontologies w.r.t. OWL 2 Full. However, some successes were obtained for ALCO Full. Since ALCO Full is an undecidable sub-language of OWL 2 Full, there is hope that the failures were not due to undecidability but rather due to the large number of axioms. This needs to be investigated further. Model finding for RDFS worked quite efficiently, which is interesting because we do not know of any tool that detects RDFS non-entailments.

In the future we plan to extend the approach to datatype reasoning, which is of high practical relevance in the Semantic Web. It may be possible to take advantage of the typed first-order or typed higher-order form of the TPTP language to effectively encode the datatypes, and reason using ATP systems that take advantage of the type information. Another topic for further research is to develop techniques for identifying parts of the FOL axiomatization that are relevant to a given reasoning task. It is hoped that by taking into account OWL 2 Full specific knowledge, more precise axiom selection than offered by the generic SInE approach will be possible. An important area of development will be query answering, i.e., the ability to obtain explicit answers to users' questions. For future OWL 2 Full reasoners this will be a very relevant reasoning task, particularly with respect to the current extension of the standard RDF query language SPARQL towards "entailment regimes" (`http://www.w3.org/TR/sparql11-entailment`). This topic is also of growing interest in the ATP community, with a proposal being considered for expressing questions and answers in the TPTP language (`http://www.tptp.org/TPTP/Proposals/AnswerExtraction.html`).

References

1. Baader, F., McGuinness, D.L., Nardi, D., Patel-Schneider, P.F. (eds.): The Description Logic Handbook: Theory, Implementation, and Applications. Cambridge University Press, Cambridge (2003)
2. Fikes, R., McGuinness, D., Waldinger, R.: A First-Order Logic Semantics for Semantic Web Markup Languages. Tech. Rep. KSL-02-01, Knowledge Systems Laboratory, Stanford University, Stanford, CA 94305 (January 2002)
3. Hawke, S.: Surnia. Homepage (2003), http://www.w3.org/2003/08/surnia
4. Hayes, P.: Translating Semantic Web Languages into Common Logic. Tech. rep., IHMC Florida Institute for Human & Machine Cognition, 40 South Alcaniz Street, Pensacola, FL 32502 (18 July 2005), access: http://www.ihmc.us/users/phayes/CL/SW2SCL.html
5. Hayes, P. (ed.): RDF Semantics. W3C Recommendation (10 February 2004), access: http://www.w3.org/TR/rdf-mt/
6. Horrocks, I., Voronkov, A.: Reasoning Support for Expressive Ontology Languages Using a Theorem Prover. In: Proceedings of the 4th International Symposium on Foundations of Information and Knowledge Systems (FoIKS 2006). LNCS, vol. 3861, pp. 201–218. Springer (2006)
7. Klyne, G., Carroll, J.J. (eds.): Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation (10 February 2004), access: http://www.w3.org/TR/rdf-concepts/
8. Motik, B.: On the Properties of Metamodeling in OWL. Journal of Logic and Computation 17(4), 617–637 (2007)
9. Motik, B., Grau, B.C., Horrocks, I., Wu, Z., Fokoue, A., Lutz, C. (eds.): OWL 2 Web Ontology Language: Profiles. W3C Recommendation (27 October 2009), access: http://www.w3.org/TR/owl2-profiles/
10. Motik, B., Patel-Schneider, P.F., Parsia, B. (eds.): OWL 2 Web Ontology Language: Structural Specification and Functional-Style Syntax. W3C Recommendation (27 October 2009), access: http://www.w3.org/TR/owl2-syntax/
11. Plaisted, D., Yahya, A.: A Relevance Restriction Strategy for Automated Deduction. Artificial Intelligence 144(1-2), 59–93 (2003)
12. Schneider, M. (ed.): OWL 2 Web Ontology Language: RDF-Based Semantics. W3C Recommendation (27 October 2009), access: http://www.w3.org/TR/owl2-rdf-based-semantics/
13. Schneider, M., Mainzer, K.: A Conformance Test Suite for the OWL 2 RL/RDF Rules Language and the OWL 2 RDF-Based Semantics. In: Hoekstra, R., Patel-Schneider, P.F. (eds.) Proceedings of the 6th International Workshop on OWL: Experiences and Directions (OWLED 2009). CEUR Workshop Proceedings, vol. 529 (2009), access: http://ceur-ws.org/Vol-529/owled2009_submission_19.pdf
14. Sutcliffe, G.: The TPTP Problem Library and Associated Infrastructure. The FOF and CNF Parts, v3.5.0. Journal of Automated Reasoning 43(4), 337–362 (2009)
15. Tsarkov, D., Riazanov, A., Bechhofer, S., Horrocks, I.: Using Vampire to Reason with OWL. In: Proceedings of the Third International Semantic Web Conference (ISWC 2004). LNCS, vol. 3298, pp. 471–485. Springer (2004)
16. W3C OWL Working Group (ed.): OWL 2 Web Ontology Language: Document Overview. W3C Recommendation (27 October 2009), access: http://www.w3.org/TR/owl2-overview/
17. W3C WebOnt OWL Working Group: OWL 1 Test Results (9 March 2004), http://www.w3.org/2003/08/owl-systems/test-results-out

A Detailed Raw Result Data

This appendix provides detailed raw result data that underlies the experimental results reported in Section 5. The data is given in tables, which present the names of the test cases in the first column, and the results of individual reasoning experiments in the other columns. The result of each experiment consists of one of the possible outcomes defined at the beginning of Section 5 and, optionally, the duration of the reasoning experiment, given as the number of seconds it took. All result data presented here is also available in electronic form as part of the *supplementary material* for this paper (see the download link at the beginning of Section 4).

A.1 Language Coverage Results

The following tables provide the raw result data that underlies the aggregated results for the language coverage experiments, as reported in Section 5.1. Table 7 contains the result data that was obtained from evaluating the Semantic Web reasoners and from the FOL theorem provers when used with the complete OWL 2 Full axiomatization; the corresponding aggregated results were reported in Tables 2 and 1, respectively. The remaining tables provide the raw results from testing the FOL reasoners on those test cases where they had failed originally, now using small but sufficient subaxiomatizations that were manually crafted for each of the test cases.

Table 7: Result data of the language coverage experiments for Semantic Web reasoners and for FOL theorem provers when used with the complete OWL 2 Full axiomatization. Time values have only be measured for the FOL reasoners. PE=Pellet, HE=HermiT, FA=FaCT++, BO=BigOWLIM, JE=Jena, PA=Parliament, VA=Vampire, IS=iProver-SInE

Test Case	PE	HE	FA	BO	JE	PA	VA	IS
rdfbased-sem-bool-complement-data	-	-	?	-	-	-	+ (0.07)	+ (0.16)
rdfbased-sem-bool-complement-ext	+	+	+	-	-	-	+ (0.33)	+ (0.61)
rdfbased-sem-bool-complement-inst	+	+	+	+	-	?	+ (0.28)	+ (0.29)
rdfbased-sem-bool-demorgan	+	+	+	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-bool-intersection-data-localize	+	+	?	-	-	-	+ (0.60)	+ (38.46)
rdfbased-sem-bool-intersection-ext	+	+	+	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-bool-intersection-inst-comp	+	+	+	+	+	-	+ (0.83)	+ (69.73)
rdfbased-sem-bool-intersection-inst-expr	+	+	+	+	+	-	+ (0.60)	+ (74.89)
rdfbased-sem-bool-intersection-localize	+	+	+	+	+	-	+ (0.87)	+ (73.54)
rdfbased-sem-bool-intersection-term	+	+	+	+	+	-	+ (1.07)	+ (81.20)
rdfbased-sem-bool-tollens	+	+	+	-	-	-	+ (0.41)	+ (0.57)
rdfbased-sem-bool-union-data-localize	+	+	?	-	-	-	+ (0.54)	+ (79.34)
rdfbased-sem-bool-union-ext	+	+	+	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-bool-union-inst-comp	+	+	+	+	+	-	+ (0.71)	+ (7.52)
rdfbased-sem-bool-union-inst-expr	+	+	+	-	-	-	? (300.00)	+ (74.41)
rdfbased-sem-bool-union-localize	+	+	+	+	+	-	+ (0.85)	+ (79.08)
rdfbased-sem-bool-union-term	+	+	+	+	+	-	+ (1.10)	+ (82.89)
rdfbased-sem-chain-def	+	+	+	+	-	-	+ (0.50)	+ (75.35)
rdfbased-sem-chain-ext	?	+	+	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-chain-localize	+	+	+	-	-	-	+ (0.50)	+ (31.96)
rdfbased-sem-chain-subprop	?	+	+	-	-	-	? (300.00)	+ (7.58)
rdfbased-sem-char-asymmetric-ext	+	+	+	-	-	-	+ (0.36)	+ (108.43)
rdfbased-sem-char-asymmetric-inst	+	+	+	+	-	?	+ (0.19)	+ (0.09)
rdfbased-sem-char-asymmetric-term	+	+	+	+	-	?	+ (0.24)	+ (0.09)
rdfbased-sem-char-functional-ext	+	+	+	-	+	-	+ (0.36)	+ (108.24)
rdfbased-sem-char-functional-inst	+	+	+	+	+	+	+ (0.37)	+ (0.40)
rdfbased-sem-char-inversefunc-data	+	+	-	+	+	+	+ (0.38)	+ (0.40)
rdfbased-sem-char-inversefunc-ext	+	+	+	-	+	-	+ (0.47)	+ (109.98)
rdfbased-sem-char-inversefunc-inst	+	+	+	+	+	+	+ (0.41)	+ (0.39)
rdfbased-sem-char-inversefunc-term	+	+	+	-	+	-	+ (1.08)	+ (6.65)
rdfbased-sem-char-irreflexive-ext	+	+	+	-	-	-	+ (0.44)	+ (78.75)
rdfbased-sem-char-irreflexive-inst	+	+	+	+	-	?	+ (0.03)	+ (0.08)
rdfbased-sem-char-irreflexive-term	+	+	+	-	-	?	+ (0.03)	+ (0.17)
rdfbased-sem-char-reflexive-ext	-	-	?	-	-	-	+ (0.40)	+ (10.01)

rdfbased-sem-char-reflexive-inst	+	+	+	-	-	-	+ (0.04)	+ (0.15)
rdfbased-sem-char-symmetric-ext	+	+	+	-	-	-	+ (0.82)	+ (131.46)
rdfbased-sem-char-symmetric-inst	+	+	+	+	+	+	+ (0.18)	+ (0.09)
rdfbased-sem-char-transitive-ext	+	+	+	-	-	-	? (300.00)	+ (131.03)
rdfbased-sem-char-transitive-inst	+	+	+	+	+	+	+ (0.39)	+ (0.11)
rdfbased-sem-char-transitive-term	-	?	?	-	-	-	+ (0.41)	+ (2.30)
rdfbased-sem-class-alldifferent-ext	-	-	?	+	-	-	+ (0.03)	+ (0.16)
rdfbased-sem-class-alldifferent-type	+	+	+	+	-	-	+ (0.02)	+ (0.17)
rdfbased-sem-class-alldisjointclasses-ext	-	-	?	+	-	-	+ (0.03)	+ (0.10)
rdfbased-sem-class-alldisjointclasses-type	+	+	+	+	-	-	+ (0.02)	+ (0.11)
rdfbased-sem-class-alldisjointproperties-ext	-	-	?	+	-	-	+ (0.03)	+ (0.10)
rdfbased-sem-class-alldisjointproperties-type	+	+	+	+	-	-	+ (0.02)	+ (0.10)
rdfbased-sem-class-annotation-ext	-	-	?	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-annotation-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-annotationproperty-type	+	+	+	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-asymmetricproperty-ext	-	-	?	+	-	-	+ (0.37)	+ (0.18)
rdfbased-sem-class-asymmetricproperty-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-axiom-ext	-	-	?	+	-	-	+ (0.03)	+ (0.09)
rdfbased-sem-class-axiom-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-datarange-ext	-	-	?	+	-	-	+ (0.32)	+ (0.16)
rdfbased-sem-class-datarange-type	+	+	+	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-datatype-ext	-	-	?	+	-	-	+ (0.28)	+ (0.16)
rdfbased-sem-class-datatype-type	+	+	+	+	+	-	+ (0.04)	+ (0.08)
rdfbased-sem-class-datatypeproperty-type	+	+	+	+	+	-	+ (0.03)	+ (0.12)
rdfbased-sem-class-deprecatedclass-ext	-	-	?	+	-	-	+ (0.32)	+ (0.15)
rdfbased-sem-class-deprecatedclass-type	+	+	+	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-deprecatedproperty-ext	-	-	?	+	-	-	+ (0.41)	+ (0.18)
rdfbased-sem-class-deprecatedproperty-type	+	+	+	+	-	-	+ (0.03)	+ (0.09)
rdfbased-sem-class-functionalproperty-ext	-	-	?	+	+	-	+ (0.34)	+ (0.42)
rdfbased-sem-class-functionalproperty-type	+	+	+	+	+	-	+ (0.02)	+ (0.14)
rdfbased-sem-class-inversefunctionalproperty-ext	-	-	?	+	+	-	+ (0.41)	+ (0.39)
rdfbased-sem-class-inversefunctionalproperty-type	+	+	+	+	+	-	+ (0.02)	+ (0.15)
rdfbased-sem-class-irreflexiveproperty-ext	-	-	?	+	-	-	+ (0.40)	+ (0.18)
rdfbased-sem-class-irreflexiveproperty-type	+	+	+	+	-	-	+ (0.03)	+ (0.09)
rdfbased-sem-class-literal-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-namedindividual-ext	-	+	?	+	-	-	+ (0.36)	+ (0.10)
rdfbased-sem-class-namedindividual-type	+	+	+	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-negativepropertyassertion-ext	-	-	?	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-negativepropertyassertion-type	+	+	+	+	-	-	+ (0.03)	+ (0.09)
rdfbased-sem-class-nothing-ext	+	+	+	+	+	?	+ (0.03)	+ (0.12)
rdfbased-sem-class-nothing-term	+	+	+	+	-	-	+ (0.42)	+ (0.41)
rdfbased-sem-class-nothing-type	+	+	-	+	+	-	+ (0.03)	+ (0.10)
rdfbased-sem-class-objectproperty-ext	-	-	?	+	+	-	+ (0.33)	+ (0.18)
rdfbased-sem-class-objectproperty-type	+	+	+	+	+	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-ontology-type	+	+	+	+	+	-	+ (0.02)	+ (0.09)
rdfbased-sem-class-ontologyproperty-type	+	+	+	+	+	-	+ (0.02)	+ (0.09)
rdfbased-sem-class-owlclass-ext	-	-	?	+	+	-	+ (0.36)	+ (0.15)
rdfbased-sem-class-owlclass-type	+	+	+	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-property-ext	-	-	?	+	-	-	+ (0.38)	+ (0.18)
rdfbased-sem-class-property-type	+	+	+	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-rdfsclass-ext	-	-	?	+	-	-	+ (0.42)	+ (0.16)
rdfbased-sem-class-rdfsclass-type	+	+	+	+	+	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-reflexiveproperty-ext	-	-	?	+	-	-	+ (0.38)	+ (0.16)
rdfbased-sem-class-reflexiveproperty-type	+	+	+	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-class-resource-ext	-	+	?	+	-	-	+ (0.31)	+ (0.13)
rdfbased-sem-class-resource-type	+	+	+	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-restriction-ext	-	-	?	+	+	-	+ (0.40)	+ (0.24)
rdfbased-sem-class-restriction-type	+	+	+	+	+	-	+ (0.02)	+ (0.16)
rdfbased-sem-class-symmetricproperty-ext	-	-	?	+	+	-	+ (0.44)	+ (0.19)
rdfbased-sem-class-symmetricproperty-type	+	+	+	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-class-thing-ext	-	-	?	-	+	-	+ (0.03)	+ (0.08)
rdfbased-sem-class-thing-term	+	+	+	+	+	-	+ (0.35)	+ (0.22)
rdfbased-sem-class-thing-type	+	+	-	+	+	-	+ (0.03)	+ (0.09)
rdfbased-sem-class-transitiveproperty-ext	-	-	?	+	+	-	+ (0.32)	+ (0.18)
rdfbased-sem-class-transitiveproperty-type	+	+	+	+	+	-	+ (0.03)	+ (0.08)
rdfbased-sem-enum-data-localize	+	+	-	-	-	-	+ (0.56)	+ (108.48)
rdfbased-sem-enum-ext	+	+	+	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-enum-inst-closed	+	+	-	-	-	-	+ (0.48)	+ (107.23)
rdfbased-sem-enum-inst-included	+	+	+	+	+	-	+ (0.48)	+ (74.77)
rdfbased-sem-eqdis-different-ext	+	+	+	-	-	-	+ (0.36)	+ (1.99)
rdfbased-sem-eqdis-different-irrflxv	-	-	-	-	-	?	+ (0.04)	+ (0.14)
rdfbased-sem-eqdis-different-sameas	+	+	+	+	+	?	+ (0.30)	+ (0.15)
rdfbased-sem-eqdis-different-sym	+	+	+	-	+	-	+ (0.04)	+ (0.20)
rdfbased-sem-eqdis-disclass-eqclass	+	+	+	+	+	?	+ (0.34)	+ (0.42)
rdfbased-sem-eqdis-disclass-ext	+	+	+	-	-	-	+ (0.38)	+ (110.98)
rdfbased-sem-eqdis-disclass-inst	+	+	+	+	+	?	+ (0.34)	+ (0.21)
rdfbased-sem-eqdis-disclass-irrflxv	-	?	-	+	+	?	+ (0.06)	+ (0.20)
rdfbased-sem-eqdis-disclass-sym	+	+	+	-	+	-	+ (0.41)	+ (0.09)
rdfbased-sem-eqdis-disjointunion-composite	?	+	+	-	-	-	? (300.00)	+ (136.00)
rdfbased-sem-eqdis-disjointunion-disjoint	+	+	+	-	-	-	+ (0.85)	+ (79.84)
rdfbased-sem-eqdis-disjointunion-localize	+	+	+	-	-	-	+ (1.13)	+ (39.35)
rdfbased-sem-eqdis-disjointunion-union	+	+	+	-	-	-	? (300.00)	+ (116.83)
rdfbased-sem-eqdis-disprop-eqprop	+	+	+	+	-	?	+ (0.34)	+ (0.09)
rdfbased-sem-eqdis-disprop-ext	+	+	+	-	-	-	+ (0.49)	+ (108.55)

rdfbased-sem-eqdis-disprop-inst	+	+	+	+	-	?	+ (0.31)	+ (0.07)
rdfbased-sem-eqdis-disprop-irrflxv	-	-	-	+	-	?	+ (0.27)	+ (0.08)
rdfbased-sem-eqdis-disprop-sym	+	+	+	-	-	-	+ (0.41)	+ (0.09)
rdfbased-sem-eqdis-eqclass-ext	+	+	+	-	+	-	+ (1.80)	+ (30.22)
rdfbased-sem-eqdis-eqclass-inst	+	+	+	+	+	-	+ (0.41)	+ (0.28)
rdfbased-sem-eqdis-eqclass-rflxv	+	+	+	+	+	-	+ (0.42)	+ (0.57)
rdfbased-sem-eqdis-eqclass-subclass-1	+	+	+	+	+	-	+ (0.41)	+ (0.68)
rdfbased-sem-eqdis-eqclass-subclass-2	+	+	+	+	+	-	+ (0.45)	+ (0.60)
rdfbased-sem-eqdis-eqclass-subst	+	+	+	+	-	-	+ (0.44)	+ (3.19)
rdfbased-sem-eqdis-eqclass-sym	+	+	+	+	+	-	+ (0.41)	+ (0.33)
rdfbased-sem-eqdis-eqclass-trans	+	+	+	+	+	-	+ (0.41)	+ (1.89)
rdfbased-sem-eqdis-eqprop-ext	+	+	+	-	-	-	? (300.00)	+ (116.80)
rdfbased-sem-eqdis-eqprop-inst	+	+	+	+	+	-	+ (0.33)	+ (0.07)
rdfbased-sem-eqdis-eqprop-rflxv	+	+	+	+	-	-	+ (0.43)	+ (0.72)
rdfbased-sem-eqdis-eqprop-subprop-1	+	+	+	+	+	-	+ (0.45)	+ (0.37)
rdfbased-sem-eqdis-eqprop-subprop-2	+	+	+	+	+	-	? (300.00)	+ (0.58)
rdfbased-sem-eqdis-eqprop-subst	+	+	+	+	+	-	+ (0.46)	+ (0.75)
rdfbased-sem-eqdis-eqprop-sym	+	+	+	+	+	-	+ (0.43)	+ (0.10)
rdfbased-sem-eqdis-eqprop-trans	+	+	+	+	+	-	? (300.00)	+ (0.13)
rdfbased-sem-eqdis-sameas-ext	+	+	?	-	-	-	+ (0.43)	+ (10.88)
rdfbased-sem-eqdis-sameas-rflxv	+	+	+	+	-	-	+ (0.04)	+ (0.51)
rdfbased-sem-eqdis-sameas-subst	-	-	?	+	+	-	+ (0.34)	+ (0.53)
rdfbased-sem-eqdis-sameas-sym	+	+	?	+	+	-	+ (0.05)	+ (0.29)
rdfbased-sem-eqdis-sameas-trans	+	+	?	+	+	-	+ (0.09)	+ (0.62)
rdfbased-sem-facet-def	?	?	?	-	-	-	? (300.00)	? (285.37)
rdfbased-sem-facet-empty	?	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-facet-localize	-	-	-	-	-	-	? (285.09)	? (300.00)
rdfbased-sem-facet-sub	-	-	?	-	-	-	? (300.00)	? (182.76)
rdfbased-sem-facet-unknown	-	-	-	-	-	?	? (300.00)	? (235.41)
rdfbased-sem-inv-ext	+	+	+	-	-	-	? (300.00)	+ (278.71)
rdfbased-sem-inv-inst	+	+	+	+	+	+	+ (0.35)	+ (0.07)
rdfbased-sem-inv-sym	+	+	+	-	+	-	+ (0.34)	+ (0.10)
rdfbased-sem-inv-trans	+	+	+	-	-	-	? (300.00)	+ (1.05)
rdfbased-sem-key-def	+	+	?	+	-	-	? (300.00)	+ (74.85)
rdfbased-sem-key-ext	?	+	?	-	-	-	? (300.00)	? (284.78)
rdfbased-sem-key-localize	+	+	?	-	-	-	+ (0.56)	+ (74.17)
rdfbased-sem-ndis-alldifferent-bw	+	+	+	-	-	-	+ (0.44)	+ (82.32)
rdfbased-sem-ndis-alldifferent-bw-distinctmembers	+	+	+	-	-	-	+ (0.92)	+ (80.11)
rdfbased-sem-ndis-alldifferent-fw	-	-	-	+	-	?	+ (0.40)	+ (72.50)
rdfbased-sem-ndis-alldifferent-fw-distinctmembers	+	+	+	-	+	?	+ (0.56)	+ (67.88)
rdfbased-sem-ndis-alldisjointclasses-bw	+	+	+	-	-	-	? (300.00)	+ (74.11)
rdfbased-sem-ndis-alldisjointclasses-fw	+	+	+	+	-	?	+ (0.41)	+ (73.69)
rdfbased-sem-ndis-alldisjointclasses-localize	+	+	+	+	-	-	+ (0.39)	+ (36.59)
rdfbased-sem-ndis-alldisjointproperties-bw	-	?	?	-	-	-	? (300.00)	+ (73.86)
rdfbased-sem-ndis-alldisjointproperties-fw	+	+	+	+	-	?	+ (0.57)	+ (72.29)
rdfbased-sem-ndis-alldisjointproperties-localize	+	+	+	+	-	-	+ (0.46)	+ (13.37)
rdfbased-sem-npa-dat-bw	+	+	+	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-npa-dat-dnpa	+	+	+	-	-	-	? (300.00)	? (214.25)
rdfbased-sem-npa-dat-fw	-	-	-	+	-	?	+ (0.41)	+ (0.17)
rdfbased-sem-npa-dat-localize	+	+	-	-	-	-	+ (0.31)	+ (0.56)
rdfbased-sem-npa-dat-npa	+	+	+	-	-	-	+ (0.36)	+ (0.88)
rdfbased-sem-npa-ind-bw	+	+	+	-	-	-	+ (0.11)	+ (2.75)
rdfbased-sem-npa-ind-fw	-	-	-	+	-	?	+ (0.35)	+ (0.16)
rdfbased-sem-parts-annotationproperties-instance	-	-	?	-	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-parts-annotationproperties-super	+	+	+	+	-	-	+ (0.03)	+ (0.10)
rdfbased-sem-parts-classes-instance	-	-	?	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-parts-classes-super	-	-	?	+	+	-	+ (0.03)	+ (0.08)
rdfbased-sem-parts-datatypeproperties-instance	-	-	?	-	-	-	+ (0.05)	+ (0.66)
rdfbased-sem-parts-datatypeproperties-super	+	+	+	+	+	-	+ (0.03)	+ (0.14)
rdfbased-sem-parts-datatypes-instance	-	-	?	-	-	-	+ (0.03)	+ (0.35)
rdfbased-sem-parts-datatypes-super	+	+	?	+	+	-	+ (0.02)	+ (0.15)
rdfbased-sem-parts-individuals-nonempty	-	-	?	-	-	-	+ (0.03)	+ (0.07)
rdfbased-sem-parts-literals-super	-	-	?	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-parts-ontologies-super	-	-	?	+	+	-	+ (0.03)	+ (0.07)
rdfbased-sem-parts-ontologyproperties-instance	+	+	+	-	-	-	+ (0.16)	+ (0.66)
rdfbased-sem-parts-ontologyproperties-super	+	+	+	+	+	-	+ (0.03)	+ (0.10)
rdfbased-sem-parts-properties-instance	-	-	?	-	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-parts-properties-super	-	-	?	+	+	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-allvaluesfrom-ext	-	-	?	+	-	-	+ (0.43)	+ (0.87)
rdfbased-sem-prop-allvaluesfrom-type	+	+	+	+	-	-	+ (0.03)	+ (0.14)
rdfbased-sem-prop-annotatedproperty-ext	-	-	?	+	-	-	+ (0.35)	+ (0.25)
rdfbased-sem-prop-annotatedproperty-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-annotatedsource-ext	-	-	?	+	-	-	+ (0.36)	+ (0.24)
rdfbased-sem-prop-annotatedsource-type	+	+	+	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-annotatedtarget-ext	-	-	?	+	-	-	+ (0.36)	+ (0.26)
rdfbased-sem-prop-annotatedtarget-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-assertionproperty-ext	-	-	?	+	-	-	+ (0.37)	+ (1.10)
rdfbased-sem-prop-assertionproperty-type	+	+	+	+	-	-	+ (0.03)	+ (0.12)
rdfbased-sem-prop-backwardcompatiblewith-ext	+	+	-	+	+	-	+ (0.42)	+ (0.44)
rdfbased-sem-prop-backwardcompatiblewith-type-annot	+	+	-	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-backwardcompatiblewith-type-onto	-	-	?	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-bottomdataproperty-ext-hi	+	+	?	+	-	-	+ (0.40)	+ (0.75)
rdfbased-sem-prop-bottomdataproperty-ext-lo	+	+	?	-	-	-	+ (0.36)	+ (0.52)
rdfbased-sem-prop-bottomdataproperty-term	-	-	?	-	-	-	+ (0.35)	+ (1.59)

rdfbased-sem-prop-bottomdataproperty-type	+	+	-	+	-	-	+ (0.02)	+ (0.12)
rdfbased-sem-prop-bottomobjectproperty-ext-hi	+	+	+	+	-	-	+ (0.32)	+ (0.24)
rdfbased-sem-prop-bottomobjectproperty-ext-lo	+	+	?	-	-	-	+ (0.39)	+ (0.40)
rdfbased-sem-prop-bottomobjectproperty-term	+	+	?	-	-	-	+ (0.35)	+ (0.18)
rdfbased-sem-prop-bottomobjectproperty-type	+	+	-	+	-	-	+ (0.03)	+ (0.12)
rdfbased-sem-prop-cardinality-ext	-	-	?	+	-	-	+ (0.58)	+ (1.61)
rdfbased-sem-prop-cardinality-type	+	+	+	+	-	-	+ (0.03)	+ (0.17)
rdfbased-sem-prop-comment-ext	+	+	-	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-comment-type	+	+	-	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-complementof-ext	-	-	?	+	-	-	+ (0.38)	+ (0.65)
rdfbased-sem-prop-complementof-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-datatypecomplementof-ext	-	-	?	+	-	-	+ (0.43)	+ (0.62)
rdfbased-sem-prop-datatypecomplementof-type	+	+	+	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-deprecated-ext	+	+	-	+	-	-	+ (0.35)	+ (0.40)
rdfbased-sem-prop-deprecated-type	+	+	-	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-differentfrom-ext	-	+	?	+	-	-	+ (0.36)	+ (0.41)
rdfbased-sem-prop-differentfrom-type	+	+	+	+	+	-	+ (0.03)	+ (0.15)
rdfbased-sem-prop-disjointunionof-ext	-	-	?	+	-	-	+ (0.35)	+ (1.59)
rdfbased-sem-prop-disjointunionof-type	+	+	+	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-prop-disjointwith-ext	-	-	?	+	+	-	+ (0.42)	+ (0.45)
rdfbased-sem-prop-disjointwith-type	+	+	+	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-distinctmembers-ext	-	-	?	+	-	-	+ (0.42)	+ (1.91)
rdfbased-sem-prop-distinctmembers-type	+	+	+	+	-	-	+ (0.03)	+ (0.15)
rdfbased-sem-prop-equivalentclass-ext	-	-	?	+	+	-	+ (0.17)	+ (0.58)
rdfbased-sem-prop-equivalentclass-type	+	+	+	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-equivalentproperty-ext	-	-	?	+	-	-	+ (0.37)	+ (0.89)
rdfbased-sem-prop-equivalentproperty-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-haskey-ext	-	-	?	+	-	-	+ (0.42)	+ (2.19)
rdfbased-sem-prop-haskey-type	+	+	+	+	-	-	+ (0.03)	+ (0.15)
rdfbased-sem-prop-hasself-ext	-	-	?	+	-	-	+ (0.41)	+ (0.65)
rdfbased-sem-prop-hasself-type	+	+	+	+	-	-	+ (0.02)	+ (0.15)
rdfbased-sem-prop-hasvalue-ext	-	-	?	+	-	-	+ (0.36)	+ (0.64)
rdfbased-sem-prop-hasvalue-type	+	+	+	+	-	-	+ (0.02)	+ (0.14)
rdfbased-sem-prop-imports-ext	-	-	?	+	+	-	+ (0.41)	+ (0.48)
rdfbased-sem-prop-imports-type	-	-	?	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-incompatiblewith-ext	+	+	-	+	+	-	+ (0.38)	+ (0.45)
rdfbased-sem-prop-incompatiblewith-type-annot	+	+	-	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-incompatiblewith-type-onto	-	-	?	+	-	-	+ (0.03)	+ (0.10)
rdfbased-sem-prop-intersectionof-ext	-	-	?	+	-	-	+ (0.43)	+ (1.08)
rdfbased-sem-prop-intersectionof-type	+	+	+	+	+	-	+ (0.03)	+ (0.10)
rdfbased-sem-prop-inverseof-ext	-	-	?	+	-	-	+ (0.37)	+ (0.89)
rdfbased-sem-prop-inverseof-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-isdefinedby-ext	+	+	-	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-isdefinedby-type	+	+	-	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-prop-label-ext	+	+	-	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-label-type	+	+	-	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-maxcardinality-ext	-	-	?	+	-	-	+ (0.49)	+ (1.14)
rdfbased-sem-prop-maxcardinality-type	+	+	+	+	-	-	+ (0.02)	+ (0.16)
rdfbased-sem-prop-maxqualifiedcardinality-ext	-	-	?	+	-	-	+ (0.58)	+ (1.33)
rdfbased-sem-prop-maxqualifiedcardinality-type	+	+	+	+	-	-	+ (0.03)	+ (0.17)
rdfbased-sem-prop-members-ext	-	-	?	+	-	-	+ (0.31)	+ (1.11)
rdfbased-sem-prop-members-type	+	+	+	+	-	-	+ (0.03)	+ (0.17)
rdfbased-sem-prop-mincardinality-ext	-	-	?	+	-	-	+ (0.58)	+ (5.87)
rdfbased-sem-prop-mincardinality-type	+	+	+	+	-	-	+ (0.03)	+ (0.17)
rdfbased-sem-prop-minqualifiedcardinality-ext	-	-	?	+	-	-	+ (0.62)	+ (6.54)
rdfbased-sem-prop-minqualifiedcardinality-type	+	+	+	+	-	-	+ (0.02)	+ (0.17)
rdfbased-sem-prop-onclass-ext	-	-	?	+	-	-	+ (0.30)	+ (1.04)
rdfbased-sem-prop-onclass-type	+	+	+	+	-	-	+ (0.03)	+ (0.16)
rdfbased-sem-prop-ondatarange-ext	-	-	?	+	-	-	+ (0.43)	+ (1.01)
rdfbased-sem-prop-ondatarange-type	+	+	+	+	-	-	+ (0.02)	+ (0.17)
rdfbased-sem-prop-ondatatype-ext	-	-	?	+	-	-	+ (0.35)	+ (0.51)
rdfbased-sem-prop-ondatatype-type	+	+	+	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-prop-oneof-ext	-	-	?	+	-	-	+ (0.33)	+ (1.01)
rdfbased-sem-prop-oneof-type	+	+	+	+	+	-	+ (0.02)	+ (0.14)
rdfbased-sem-prop-onproperty-ext	-	-	?	+	-	-	+ (0.35)	+ (1.23)
rdfbased-sem-prop-onproperty-type	+	+	+	+	+	-	+ (0.02)	+ (0.17)
rdfbased-sem-prop-priorversion-ext	+	+	-	+	+	-	+ (0.37)	+ (0.45)
rdfbased-sem-prop-priorversion-type-annot	+	+	-	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-prop-priorversion-type-onto	-	-	?	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-prop-propertychainaxiom-ext	-	-	?	+	-	-	+ (0.33)	+ (1.54)
rdfbased-sem-prop-propertychainaxiom-type	+	+	+	+	-	-	+ (0.02)	+ (0.10)
rdfbased-sem-prop-propertydisjointwith-ext	-	-	?	+	-	-	+ (0.41)	+ (1.14)
rdfbased-sem-prop-propertydisjointwith-type	+	+	+	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-prop-qualifiedcardinality-ext	-	-	?	+	-	-	+ (0.65)	+ (1.30)
rdfbased-sem-prop-qualifiedcardinality-type	+	+	+	+	-	-	+ (0.02)	+ (0.18)
rdfbased-sem-prop-sameas-ext	-	+	?	+	-	-	+ (0.36)	+ (0.40)
rdfbased-sem-prop-sameas-type	+	+	+	+	+	-	+ (0.03)	+ (0.15)
rdfbased-sem-prop-seealso-ext	+	+	-	+	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-prop-seealso-type	+	+	-	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-prop-somevaluesfrom-ext	-	-	?	+	-	-	+ (0.37)	+ (0.83)
rdfbased-sem-prop-somevaluesfrom-type	+	+	+	+	-	-	+ (0.02)	+ (0.15)
rdfbased-sem-prop-sourceindividual-ext	-	-	?	+	-	-	+ (0.37)	+ (0.64)
rdfbased-sem-prop-sourceindividual-type	+	+	+	+	-	-	+ (0.02)	+ (0.12)
rdfbased-sem-prop-targetindividual-ext	-	-	?	+	-	-	+ (0.42)	+ (0.34)

rdfbased-sem-prop-targetindividual-type	+	+	+	+	-	-	+ (0.02)	+ (0.09)
rdfbased-sem-prop-targetvalue-ext	-	-	?	+	-	-	+ (0.44)	+ (1.08)
rdfbased-sem-prop-targetvalue-type	+	+	+	+	-	-	+ (0.03)	+ (0.12)
rdfbased-sem-prop-topdataproperty-ext-hi	+	+	?	+	-	-	+ (0.44)	+ (0.99)
rdfbased-sem-prop-topdataproperty-ext-lo	+	+	+	-	-	-	+ (0.04)	+ (0.44)
rdfbased-sem-prop-topdataproperty-term	-	-	?	-	-	-	+ (0.40)	+ (2.26)
rdfbased-sem-prop-topdataproperty-type	+	+	-	+	-	-	+ (0.02)	+ (0.11)
rdfbased-sem-prop-topobjectproperty-ext-hi	+	+	+	+	-	-	+ (0.32)	+ (0.21)
rdfbased-sem-prop-topobjectproperty-ext-lo	+	+	+	-	-	-	+ (0.04)	+ (0.11)
rdfbased-sem-prop-topobjectproperty-term	+	+	?	-	-	-	+ (0.38)	+ (0.20)
rdfbased-sem-prop-topobjectproperty-type	+	+	-	+	-	-	+ (0.03)	+ (0.14)
rdfbased-sem-prop-unionof-ext	-	-	?	+	-	-	+ (0.39)	+ (4.83)
rdfbased-sem-prop-unionof-type	+	+	+	+	-	-	+ (0.03)	+ (0.09)
rdfbased-sem-prop-versioninfo-ext	+	+	-	+	-	-	+ (0.34)	+ (0.42)
rdfbased-sem-prop-versioninfo-type	+	+	-	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-prop-versioniri-ext	-	-	?	+	-	-	+ (0.43)	+ (0.47)
rdfbased-sem-prop-versioniri-type	-	-	?	+	-	-	+ (0.03)	+ (0.09)
rdfbased-sem-prop-withrestrictions-ext	-	-	?	+	-	-	+ (0.44)	+ (1.85)
rdfbased-sem-prop-withrestrictions-type	+	+	+	+	-	-	+ (0.03)	+ (0.08)
rdfbased-sem-rdf-container-highval-axiom	+	+	+	+	-	-	? (300.00)	? (181.83)
rdfbased-sem-rdf-container-initval-axiom	+	+	+	+	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdf-list-axiom	+	+	+	+	+	-	+ (0.02)	+ (0.17)
rdfbased-sem-rdf-reify-axiom	+	+	+	+	+	-	+ (0.02)	+ (0.21)
rdfbased-sem-rdf-type-axiom	+	+	+	+	+	-	+ (0.03)	+ (0.07)
rdfbased-sem-rdf-type-cond	+	+	+	+	+	-	+ (0.03)	+ (0.19)
rdfbased-sem-rdf-value-axiom	+	+	+	+	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdf-xmlliteral-type	-	-	?	-	-	-	? (300.00)	? (181.71)
rdfbased-sem-rdfs-annotate-axiom	+	+	-	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-rdfs-class	+	+	+	+	+	-	+ (0.19)	+ (0.12)
rdfbased-sem-rdfs-container-cond	-	-	?	-	+	-	+ (0.03)	+ (0.11)
rdfbased-sem-rdfs-container-highval-axiom	-	-	?	+	-	-	? (300.00)	? (220.78)
rdfbased-sem-rdfs-container-initval-axiom	-	-	?	+	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-container-static-axiom	-	-	?	+	-	-	+ (0.02)	+ (0.08)
rdfbased-sem-rdfs-data-cond	-	-	?	-	-	-	+ (0.03)	+ (0.10)
rdfbased-sem-rdfs-datatype-axiom	-	-	?	+	+	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-domain-axiom	-	-	?	+	+	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-domain-cond	+	+	+	+	+	-	+ (0.19)	+ (0.23)
rdfbased-sem-rdfs-list-axiom	-	-	?	+	-	-	+ (0.04)	+ (0.15)
rdfbased-sem-rdfs-plain-notag-type	-	-	?	-	-	-	? (300.00)	? (196.34)
rdfbased-sem-rdfs-plain-tagged-type	-	-	?	-	-	-	? (300.00)	? (202.44)
rdfbased-sem-rdfs-range-axiom	-	-	?	+	+	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-range-cond	+	+	+	+	+	-	+ (0.20)	+ (0.21)
rdfbased-sem-rdfs-reify-axiom	-	-	?	+	-	-	+ (0.03)	+ (0.06)
rdfbased-sem-rdfs-resource	-	-	?	-	-	-	+ (0.02)	+ (0.14)
rdfbased-sem-rdfs-subclass-axiom	-	-	?	+	+	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-subclass-cond	+	+	+	+	+	+	+ (0.34)	+ (0.35)
rdfbased-sem-rdfs-subclass-resource	-	-	?	-	+	-	+ (0.03)	+ (0.10)
rdfbased-sem-rdfs-subclass-rflxv	-	+	?	+	+	-	+ (0.03)	+ (0.11)
rdfbased-sem-rdfs-subclass-trans	+	+	+	+	-	+	+ (0.33)	+ (0.10)
rdfbased-sem-rdfs-subprop-axiom	-	-	?	+	+	-	+ (0.02)	+ (0.08)
rdfbased-sem-rdfs-subprop-cond	+	+	+	+	+	+	+ (0.33)	+ (0.10)
rdfbased-sem-rdfs-subprop-rflxv	-	+	?	+	+	-	+ (0.02)	+ (0.10)
rdfbased-sem-rdfs-subprop-trans	+	+	+	+	+	+	+ (0.33)	+ (0.09)
rdfbased-sem-rdfs-type-axiom	-	-	?	+	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-value-axiom	-	-	?	+	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-xmlliteral-axiom-type	+	+	+	+	+	-	+ (0.02)	+ (0.07)
rdfbased-sem-rdfs-xmlliteral-axiom-value	-	-	?	+	-	-	+ (0.02)	+ (0.06)
rdfbased-sem-rdfs-xmlliteral-illtyped	-	?	?	-	-	?	? (300.00)	? (300.00)
rdfbased-sem-rdfsext-domain-ext	+	+	+	-	-	-	+ (1.11)	+ (65.94)
rdfbased-sem-rdfsext-domain-subprop	+	+	+	+	+	-	+ (0.25)	+ (2.40)
rdfbased-sem-rdfsext-domain-superclass	+	+	+	+	+	-	+ (0.39)	+ (39.92)
rdfbased-sem-rdfsext-range-ext	+	+	+	-	+	-	? (300.00)	+ (43.53)
rdfbased-sem-rdfsext-range-subprop	+	+	+	+	+	-	+ (0.43)	+ (6.21)
rdfbased-sem-rdfsext-range-superclass	+	+	+	+	+	-	+ (0.32)	+ (11.90)
rdfbased-sem-rdfsext-subclass-ext	+	+	+	-	-	-	+ (0.87)	+ (107.13)
rdfbased-sem-rdfsext-subprop-ext	+	+	+	-	-	-	+ (0.49)	+ (111.26)
rdfbased-sem-restrict-allvalues-cmp-class	-	-	?	+	-	-	? (300.00)	+ (56.84)
rdfbased-sem-restrict-allvalues-cmp-prop	-	-	?	+	-	-	? (300.00)	+ (73.76)
rdfbased-sem-restrict-allvalues-inst-obj	-	-	?	+	+	-	+ (0.77)	+ (5.21)
rdfbased-sem-restrict-allvalues-inst-subj	-	-	?	-	+	-	+ (1.05)	+ (14.89)
rdfbased-sem-restrict-exactcard-inst-obj-two	-	?	?	-	-	-	+ (0.79)	+ (1.08)
rdfbased-sem-restrict-exactcard-inst-subj-two	-	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-exactqcr-data-localize	+	+	-	-	-	-	+ (0.66)	+ (0.78)
rdfbased-sem-restrict-exactqcr-inst-obj-two	-	?	?	-	-	-	? (300.00)	+ (15.62)
rdfbased-sem-restrict-exactqcr-inst-subj-two	-	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-hasself-cmp-prop	-	-	?	-	-	-	? (300.00)	+ (12.41)
rdfbased-sem-restrict-hasself-inst-obj	-	-	?	-	-	-	+ (0.49)	+ (0.76)
rdfbased-sem-restrict-hasself-inst-subj	-	-	?	-	-	-	+ (0.82)	+ (0.46)
rdfbased-sem-restrict-hasvalue-cmp-prop	-	-	?	+	-	-	+ (27.08)	+ (2.00)
rdfbased-sem-restrict-hasvalue-inst-obj	-	-	?	+	+	-	+ (0.42)	+ (0.86)
rdfbased-sem-restrict-hasvalue-inst-subj	-	-	?	+	+	-	+ (0.67)	+ (0.40)
rdfbased-sem-restrict-maxcard-cmp-card	-	-	?	-	-	-	? (300.00)	+ (112.57)
rdfbased-sem-restrict-maxcard-cmp-prop	-	-	?	-	-	-	? (300.00)	? (285.29)
rdfbased-sem-restrict-maxcard-inst-obj-one	-	-	-	+	+	-	+ (0.92)	+ (0.85)

rdfbased-sem-restrict-maxcard-inst-obj-zero	-	-	-	+	+	?	+ (0.65)	+ (0.84)
rdfbased-sem-restrict-maxcard-inst-subj-one	-	-	?	-	+	-	? (300.00)	+ (276.12)
rdfbased-sem-restrict-maxcard-inst-subj-zero	-	-	?	-	-	-	+ (0.67)	+ (1.61)
rdfbased-sem-restrict-maxqcr-cmp-card	-	-	?	-	-	-	? (300.00)	+ (111.11)
rdfbased-sem-restrict-maxqcr-cmp-class	-	-	?	-	-	-	? (300.00)	+ (76.42)
rdfbased-sem-restrict-maxqcr-cmp-prop	-	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-maxqcr-data-localize	+	+	-	-	-	-	+ (0.65)	+ (0.77)
rdfbased-sem-restrict-maxqcr-inst-obj-one	-	-	-	+	-	-	? (300.00)	+ (74.83)
rdfbased-sem-restrict-maxqcr-inst-obj-zero	-	-	-	+	-	?	+ (0.81)	+ (63.29)
rdfbased-sem-restrict-maxqcr-inst-subj-one	-	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-maxqcr-inst-subj-zero	-	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-mincard-cmp-card	-	-	?	-	-	-	? (300.00)	+ (0.67)
rdfbased-sem-restrict-mincard-cmp-prop	-	-	?	-	-	-	? (300.00)	+ (8.93)
rdfbased-sem-restrict-mincard-inst-obj-one	-	-	?	-	-	-	+ (0.43)	+ (0.70)
rdfbased-sem-restrict-mincard-inst-subj-one	-	-	?	-	+	-	+ (0.67)	+ (0.42)
rdfbased-sem-restrict-minqcr-cmp-card	-	-	?	-	-	-	? (300.00)	+ (74.69)
rdfbased-sem-restrict-minqcr-cmp-class	-	-	?	-	-	-	? (300.00)	+ (72.80)
rdfbased-sem-restrict-minqcr-cmp-prop	-	-	?	-	-	-	? (300.00)	+ (2.05)
rdfbased-sem-restrict-minqcr-data-localize	+	+	-	-	-	-	+ (0.60)	+ (0.70)
rdfbased-sem-restrict-minqcr-inst-obj-one	-	-	?	-	-	-	+ (0.76)	+ (10.16)
rdfbased-sem-restrict-minqcr-inst-subj-one	-	-	?	-	-	-	+ (0.79)	+ (3.96)
rdfbased-sem-restrict-somevalues-cmp-class	-	-	?	+	-	-	? (300.00)	+ (2.86)
rdfbased-sem-restrict-somevalues-cmp-prop	-	-	?	+	-	-	? (300.00)	+ (74.43)
rdfbased-sem-restrict-somevalues-inst-obj	-	-	?	-	-	-	+ (0.37)	+ (3.47)
rdfbased-sem-restrict-somevalues-inst-subj	-	-	?	+	+	-	+ (0.78)	+ (1.19)
rdfbased-sem-restrict-term-cardqcr	-	-	?	-	-	-	? (300.00)	+ (145.98)
rdfbased-sem-restrict-term-dataqcr	-	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-term-minmaxexact	-	-	?	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-term-minmaxthing	-	-	-	-	-	-	? (300.00)	? (300.00)
rdfbased-sem-restrict-term-sameall	-	-	?	-	-	-	? (300.00)	+ (74.84)
rdfbased-sem-restrict-term-selfsome	-	-	?	-	-	-	+ (27.57)	+ (1.20)
rdfbased-sem-restrict-term-somehas	-	-	?	-	-	-	? (300.00)	+ (118.06)
rdfbased-sem-restrict-term-someqcr	-	-	?	-	-	-	? (300.23)	+ (11.39)
rdfbased-sem-simple-bnode-iri	+	+	?	-	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-simple-bnode-literal	+	?	?	-	-	-	+ (0.03)	+ (0.24)
rdfbased-sem-simple-bnode-rename	+	+	?	-	-	-	+ (0.02)	+ (0.07)
rdfbased-sem-simple-bnode-same	+	+	?	-	-	-	+ (0.03)	+ (0.07)
rdfbased-sem-simple-emptygraph-any	+	+	+	+	+	+	+ (0.03)	+ (0.05)
rdfbased-sem-simple-emptygraph-self	+	+	+	+	+	+	+ (0.01)	+ (0.07)
rdfbased-sem-simple-subgraph-any	+	+	+	+	+	+	+ (0.02)	+ (0.06)
rdfbased-sem-simple-subgraph-self	+	+	+	+	+	+	+ (0.02)	+ (0.06)

Table 8: Result data of the language coverage experiments for the FOL theorem prover *iProver-SInE*, used with the small-sufficient OWL 2 Full subaxiomatizations on those test cases where it and Vampire had failed when using the complete axiomatization.

Test Case	iProver-SInE
rdfbased-sem-bool-demorgan	+ (36.39)
rdfbased-sem-bool-intersection-ext	+ (0.64)
rdfbased-sem-bool-union-ext	+ (0.54)
rdfbased-sem-chain-ext	+ (35.00)
rdfbased-sem-enum-ext	+ (0.17)
rdfbased-sem-restrict-exactcard-inst-subj-two	+ (55.83)
rdfbased-sem-restrict-exactqcr-inst-subj-two	? (300.00)
rdfbased-sem-restrict-maxcard-cmp-prop	+ (0.18)
rdfbased-sem-restrict-maxqcr-cmp-prop	+ (0.30)
rdfbased-sem-restrict-maxqcr-inst-subj-one	+ (1.30)
rdfbased-sem-restrict-maxqcr-inst-subj-zero	+ (2.34)
rdfbased-sem-restrict-term-dataqcr	+ (0.59)
rdfbased-sem-restrict-term-minmaxexact	+ (39.47)
rdfbased-sem-restrict-term-minmaxthing	+ (2.03)

Table 9: Result data of the language coverage experiments for the FOL theorem prover *Vampire*, used with the small-sufficient OWL 2 Full subaxiomatizations on those test cases where iProver-SInE failed when using the small-sufficient axiomatizations.

Test Case	Vampire
rdfbased-sem-restrict-exactqcr-inst-subj-two	+ (2.90)

A.2 OWL 2 Full-Characteristic Conclusions and Scalability Results

The following tables provide the raw result data that underlies the results for the scalability experiments, as reported in Section 5.3. All experiments were conducted using the test suite of characteristic OWL 2 Full conclusions, as introduced in Section 4.2 (see Appendix B for more detailed information about the test suite). There is one table per combination of a FOL reasoner (*iProver-SInE*, *Vampire* in auto mode, and *Vampire* using the *SInE* strategy) and either the complete OWL 2 Full axiomatization or the small-sufficient subaxiomatizations for the different test cases. While Section 5.3 lists only the results for bulk RDF data of size 1 million triples, the tables here also show results for several intermediate sizes: 1200, 10,000, and 100,000 triples. In addition, the results for no bulk data (0 triples) are presented, which were the base for the results reported in Section 5.2 for the test suite of characteristic OWL 2 Full conclusions. No result data for the characteristic conclusion tests is given here for the Semantic Web reasoners, since the data provided in Section 5.2 is already complete for them. The first column of each table gives the name of the test case, and the remaining columns gives the results for the different bulk data sizes.

Table 10: Scalability results for the theorem prover *iProver-SInE* using the complete OWL 2 Full axiomatization.

Test Case	**0**	**1200**	**10k**	**100k**	**1M**
001_Subgraph_Entailment	+ (0.15)	+ (0.17)	+ (0.34)	+ (2.16)	+ (21.98)
002_Existential_Blank_Nodes	+ (0.08)	+ (0.10)	+ (0.27)	+ (2.15)	+ (21.81)
003_Blank_Nodes_for_Literals	+ (0.08)	+ (0.10)	+ (0.27)	+ (2.06)	+ (22.51)
004_Axiomatic_Triples	+ (1.23)	+ (1.27)	+ (1.44)	+ (3.43)	+ (23.31)
005_Everything_is_a_Resource	+ (3.03)	+ (3.03)	+ (2.56)	+ (5.19)	+ (25.07)
006_Literal_Values_represented_by_URIs_and_Blank_Nodes	+ (11.66)	+ (11.65)	+ (11.77)	+ (13.61)	+ (34.00)
007_Equal_Classes	+ (74.40)	+ (61.38)	+ (50.34)	+ (76.84)	+ (96.51)
008_Inverse_Functional_Data_Properties	+ (0.41)	+ (0.42)	+ (0.59)	+ (2.41)	+ (22.66)
009_Existential_Restriction_Entailments	+ (2.35)	+ (2.35)	+ (2.48)	+ (4.36)	+ (24.15)
010_Negative_Property_Assertions	+ (89.45)	+ (91.03)	+ (90.05)	+ (92.60)	+ (111.97)
011_Entity_Types_as_Classes	+ (0.30)	+ (0.32)	+ (0.49)	+ (2.30)	+ (22.20)
012_Template_Class	? (300.00)	+ (144.50)	? (300.00)	? (300.00)	? (300.00)
013_Cliques	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
014_Harry_belongs_to_some_Species	+ (33.65)	+ (41.90)	+ (32.35)	+ (54.93)	+ (98.11)
015_Reflective_Tautologies_I	+ (0.16)	+ (0.17)	+ (0.34)	+ (2.21)	+ (22.06)
016_Reflective_Tautologies_II	+ (0.95)	+ (0.96)	+ (1.12)	+ (2.95)	+ (22.83)
017_Builtin_Based_Definitions	+ (5.31)	+ (5.19)	+ (5.36)	+ (20.17)	+ (34.27)
018_Modified_Logical_Vocabulary_Semantics	+ (0.72)	+ (0.74)	+ (0.92)	+ (2.73)	+ (22.66)
019_Disjoint_Annotation_Properties	+ (0.19)	+ (0.22)	+ (0.38)	+ (2.25)	+ (22.06)
020_Logical_Complications	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
021_Composite_Enumerations	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
022_List_Member_Access	+ (123.01)	+ (122.19)	+ (122.41)	+ (123.19)	+ (143.69)
023_Unique_List_Components	+ (4.12)	+ (4.07)	+ (4.26)	+ (6.02)	+ (25.88)
024_Cardinality_Restrictions_on_Complex_Properties	+ (14.89)	+ (14.06)	+ (13.84)	+ (16.78)	+ (37.30)
025_Cyclic_Dependencies_between_Complex_Properties	+ (117.92)	+ (118.07)	+ (120.24)	+ (120.02)	+ (136.68)
026_Inferred_Property_Characteristics_I	+ (111.18)	+ (63.40)	+ (109.99)	+ (113.43)	+ (130.79)
027_Inferred_Property_Characteristics_II	+ (122.01)	+ (120.51)	+ (121.49)	+ (120.86)	+ (143.95)
028_Inferred_Property_Characteristics_III	+ (3.56)	+ (3.58)	+ (3.66)	+ (5.59)	+ (25.48)
029_Ex_Falso_Quodlibet	+ (74.35)	+ (74.62)	+ (74.86)	+ (76.59)	+ (96.43)
030_Bad_Class	+ (18.07)	+ (18.42)	+ (18.79)	+ (25.70)	+ (45.60)
031_Large_Universe	+ (42.44)	+ (39.85)	+ (51.68)	+ (53.45)	+ (99.83)
032_Datatype_Relationships	+ (2.05)	+ (2.04)	+ (2.23)	+ (4.08)	+ (23.86)

Table 11: Scalability results for the theorem prover *Vampire* (auto mode) using the complete OWL 2 Full axiomatization.

Test Case	0	1200	10k	100k	1M
001_Subgraph_Entailment	+ (0.02)	+ (0.04)	+ (0.21)	+ (2.31)	+ (71.14)
002_Existential_Blank_Nodes	+ (0.02)	+ (0.03)	+ (0.21)	+ (2.35)	+ (216.63)
003_Blank_Nodes_for_Literals	+ (0.02)	+ (0.04)	+ (0.21)	+ (1.90)	+ (73.91)
004_Axiomatic_Triples	+ (0.43)	+ (0.46)	+ (0.67)	+ (2.49)	? (300.00)
005_Everything_is_a_Resource	+ (0.03)	+ (0.05)	+ (0.22)	+ (2.38)	? (300.00)
006_Literal_Values_represented_by_URIs_and_Blank_Nodes	+ (0.18)	+ (0.21)	+ (0.38)	+ (2.34)	? (300.00)
007_Equal_Classes	+ (0.35)	+ (0.40)	+ (0.54)	+ (2.41)	? (300.00)
008_Inverse_Functional_Data_Properties	+ (0.40)	+ (0.44)	+ (0.56)	+ (2.45)	? (300.00)
009_Existential_Restriction_Entailments	+ (0.39)	+ (0.40)	+ (0.57)	+ (2.34)	? (284.91)
010_Negative_Property_Assertions	? (285.53)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
011_Entity_Types_as_Classes	+ (0.18)	+ (0.20)	+ (0.38)	+ (2.30)	? (300.00)
012_Template_Class	? (285.57)	? (300.00)	? (284.78)	? (300.00)	? (213.28)
013_Cliques	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
014_Harry_belongs_to_some_Species	+ (1.15)	+ (1.18)	+ (1.33)	+ (2.97)	? (300.57)
015_Reflective_Tautologies_I	+ (0.03)	+ (0.05)	+ (0.21)	+ (1.99)	+ (223.44)
016_Reflective_Tautologies_II	+ (0.56)	+ (0.57)	+ (0.73)	+ (2.35)	? (300.00)
017_Builtin_Based_Definitions	+ (0.38)	+ (0.43)	+ (0.59)	+ (2.20)	? (300.22)
018_Modified_Logical_Vocabulary_Semantics	+ (0.16)	+ (0.17)	+ (0.34)	+ (2.25)	? (301.90)
019_Disjoint_Annotation_Properties	+ (0.43)	+ (0.46)	+ (0.61)	+ (2.17)	? (300.00)
020_Logical_Complications	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
021_Composite_Enumerations	? (300.00)	? (300.00)	? (300.00)	? (300.69)	? (300.00)
022_List_Member_Access	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
023_Unique_List_Components	+ (0.46)	+ (0.51)	+ (0.63)	+ (2.24)	? (300.00)
024_Cardinality_Restrictions_on_Complex_Properties	+ (0.71)	+ (0.73)	+ (0.85)	+ (2.47)	? (300.00)
025_Cyclic_Dependencies_between_Complex_Properties	? (300.69)	? (300.31)	? (300.00)	? (300.00)	? (300.00)
026_Inferred_Property_Characteristics_I	+ (0.47)	+ (0.48)	+ (0.64)	+ (2.28)	? (300.00)
027_Inferred_Property_Characteristics_II	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
028_Inferred_Property_Characteristics_III	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.36)
029_Ex_Falso_Quodlibet	+ (0.92)	+ (0.93)	+ (1.13)	+ (2.69)	? (300.00)
030_Bad_Class	+ (0.39)	+ (0.45)	+ (0.57)	+ (2.18)	? (300.38)
031_Large_Universe	+ (0.44)	+ (0.46)	+ (0.62)	+ (2.22)	? (300.00)
032_Datatype_Relationships	+ (0.65)	+ (0.64)	+ (0.81)	+ (2.35)	? (300.46)

Table 12: Scalability results for the theorem prover *Vampire-SInE* using the complete OWL 2 Full axiomatization.

Test Case	0	1200	10k	100k	1M
001_Subgraph_Entailment	+ (0.04)	+ (0.06)	+ (0.20)	+ (1.84)	+ (20.19)
002_Existential_Blank_Nodes	+ (0.04)	+ (0.06)	+ (0.21)	+ (1.81)	+ (19.28)
003_Blank_Nodes_for_Literals	+ (0.02)	+ (0.04)	+ (0.19)	+ (1.82)	+ (19.34)
004_Axiomatic_Triples	+ (4.99)	+ (5.04)	+ (5.21)	+ (6.77)	+ (24.38)
005_Everything_is_a_Resource	+ (0.06)	+ (0.07)	+ (0.23)	+ (1.86)	+ (20.32)
006_Literal_Values_represented_by_URIs_and_Blank_Nodes	+ (9.63)	+ (9.74)	+ (9.66)	+ (11.43)	+ (29.27)
007_Equal_Classes	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
008_Inverse_Functional_Data_Properties	+ (3.13)	+ (3.14)	+ (3.37)	+ (5.06)	+ (23.54)
009_Existential_Restriction_Entailments	? (300.00)	? (301.49)	? (300.00)	? (300.00)	? (300.00)
010_Negative_Property_Assertions	? (300.26)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
011_Entity_Types_as_Classes	+ (0.08)	+ (0.10)	+ (0.25)	+ (1.87)	+ (19.47)
012_Template_Class	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
013_Cliques	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
014_Harry_belongs_to_some_Species	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
015_Reflective_Tautologies_I	+ (0.06)	+ (0.08)	+ (0.22)	+ (1.86)	+ (19.37)
016_Reflective_Tautologies_II	+ (10.82)	+ (10.61)	+ (10.93)	+ (12.94)	+ (30.77)
017_Builtin_Based_Definitions	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
018_Modified_Logical_Vocabulary_Semantics	+ (0.27)	+ (0.29)	+ (0.44)	+ (2.09)	+ (19.45)
019_Disjoint_Annotation_Properties	+ (3.12)	+ (3.22)	+ (3.31)	+ (4.96)	+ (22.95)
020_Logical_Complications	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
021_Composite_Enumerations	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
022_List_Member_Access	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
023_Unique_List_Components	+ (6.97)	+ (6.93)	+ (6.92)	+ (8.87)	+ (26.57)
024_Cardinality_Restrictions_on_Complex_Properties	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
025_Cyclic_Dependencies_between_Complex_Properties	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
026_Inferred_Property_Characteristics_I	+ (11.97)	+ (12.16)	+ (11.97)	+ (14.30)	+ (31.64)
027_Inferred_Property_Characteristics_II	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
028_Inferred_Property_Characteristics_III	? (300.00)	? (301.10)	? (300.00)	? (300.00)	? (300.00)
029_Ex_Falso_Quodlibet	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
030_Bad_Class	+ (6.33)	+ (6.24)	+ (6.41)	+ (8.29)	+ (25.92)
031_Large_Universe	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
032_Datatype_Relationships	+ (0.09)	+ (0.11)	+ (0.25)	+ (1.94)	+ (19.53)

Table 13: Scalability results for the theorem prover *iProver-SInE* using the small-sufficient OWL 2 Full subaxiomatizations.

Test Case	0	1200	10k	100k	1M
001_Subgraph_Entailment	+ (0.04)	+ (0.07)	+ (0.24)	+ (2.03)	+ (22.87)
002_Existential_Blank_Nodes	+ (0.05)	+ (0.07)	+ (0.24)	+ (2.09)	+ (21.85)
003_Blank_Nodes_for_Literals	+ (0.05)	+ (0.07)	+ (0.26)	+ (2.14)	+ (22.07)
004_Axiomatic_Triples	+ (0.10)	+ (0.12)	+ (0.30)	+ (2.12)	+ (21.97)
005_Everything_is_a_Resource	+ (0.05)	+ (0.08)	+ (0.26)	+ (2.07)	+ (22.50)
006_Literal_Values_represented_by_URIs_and_Blank_Nodes	+ (0.07)	+ (0.09)	+ (0.26)	+ (2.08)	+ (21.96)
007_Equal_Classes	+ (0.07)	+ (0.08)	+ (0.26)	+ (2.08)	+ (22.23)
008_Inverse_Functional_Data_Properties	+ (0.06)	+ (0.09)	+ (0.27)	+ (2.08)	+ (21.97)
009_Existential_Restriction_Entailments	+ (0.05)	+ (0.08)	+ (0.25)	+ (2.08)	+ (21.89)
010_Negative_Property_Assertions	+ (0.29)	+ (0.32)	+ (0.49)	+ (2.33)	+ (22.21)
011_Entity_Types_as_Classes	+ (0.05)	+ (0.07)	+ (0.26)	+ (2.08)	+ (21.93)
012_Template_Class	+ (0.27)	+ (0.24)	+ (0.41)	+ (2.25)	+ (22.13)
013_Cliques	+ (164.20)	+ (191.29)	? (256.09)	? (259.50)	? (300.00)
014_Harry_belongs_to_some_Species	+ (0.08)	+ (0.09)	+ (0.27)	+ (2.08)	+ (21.85)
015_Reflective_Tautologies_I	+ (0.05)	+ (0.07)	+ (0.25)	+ (2.10)	+ (21.73)
016_Reflective_Tautologies_II	+ (0.11)	+ (0.13)	+ (0.30)	+ (2.12)	+ (21.76)
017_Builtin_Based_Definitions	+ (0.08)	+ (0.10)	+ (0.27)	+ (2.09)	+ (21.93)
018_Modified_Logical_Vocabulary_Semantics	+ (0.05)	+ (0.07)	+ (0.24)	+ (2.07)	+ (22.50)
019_Disjoint_Annotation_Properties	+ (0.05)	+ (0.08)	+ (0.25)	+ (2.10)	+ (21.91)
020_Logical_Complications	+ (40.67)	+ (45.04)	+ (47.79)	+ (42.16)	+ (62.69)
021_Composite_Enumerations	+ (42.32)	+ (38.32)	+ (46.15)	+ (38.11)	+ (63.10)
022_List_Member_Access	+ (0.12)	+ (0.14)	+ (0.31)	+ (2.14)	+ (22.00)
023_Unique_List_Components	+ (0.14)	+ (0.16)	+ (0.34)	+ (2.24)	+ (22.62)
024_Cardinality_Restrictions_on_Complex_Properties	+ (0.07)	+ (0.08)	+ (0.27)	+ (2.14)	+ (21.87)
025_Cyclic_Dependencies_between_Complex_Properties	+ (0.11)	+ (0.14)	+ (0.31)	+ (2.13)	+ (22.39)
026_Inferred_Property_Characteristics_I	+ (0.12)	+ (0.15)	+ (0.32)	+ (2.14)	+ (22.04)
027_Inferred_Property_Characteristics_II	+ (0.16)	+ (0.19)	+ (0.35)	+ (2.18)	+ (22.14)
028_Inferred_Property_Characteristics_III	+ (0.30)	+ (0.33)	+ (0.50)	+ (2.35)	+ (22.84)
029_Ex_Falso_Quodlibet	+ (0.09)	+ (0.11)	+ (0.28)	+ (2.11)	+ (22.51)
030_Bad_Class	+ (0.07)	+ (0.08)	+ (0.26)	+ (2.09)	+ (22.07)
031_Large_Universe	+ (0.34)	+ (0.36)	+ (0.54)	+ (2.39)	+ (22.29)
032_Datatype_Relationships	+ (0.07)	+ (0.09)	+ (0.26)	+ (2.08)	+ (21.86)

Table 14: Scalability results for the theorem prover *Vampire* (auto mode) using the small-sufficient OWL 2 Full subaxiomatizations.

Test Case	0	1200	10k	100k	1M
001_Subgraph_Entailment	+ (0.00)	+ (0.01)	+ (0.20)	+ (2.41)	+ (48.09)
002_Existential_Blank_Nodes	+ (0.00)	+ (0.05)	+ (1.89)	? (300.00)	? (301.00)
003_Blank_Nodes_for_Literals	+ (0.00)	+ (0.03)	+ (1.49)	+ (161.52)	+ (182.70)
004_Axiomatic_Triples	+ (0.01)	+ (0.06)	+ (2.02)	? (300.00)	? (300.00)
005_Everything_is_a_Resource	+ (0.00)	+ (0.08)	+ (3.17)	? (300.00)	? (300.00)
006_Literal_Values_represented_by_URIs_and_Blank_Nodes	+ (0.00)	+ (0.03)	+ (1.60)	? (300.00)	? (300.00)
007_Equal_Classes	+ (0.00)	+ (0.05)	+ (1.81)	? (300.00)	? (300.00)
008_Inverse_Functional_Data_Properties	+ (0.00)	+ (0.05)	+ (1.84)	? (300.00)	? (300.00)
009_Existential_Restriction_Entailments	+ (0.00)	+ (0.05)	+ (2.01)	? (300.00)	? (285.58)
010_Negative_Property_Assertions	+ (0.05)	+ (0.11)	+ (1.92)	? (300.00)	? (300.00)
011_Entity_Types_as_Classes	+ (0.00)	+ (0.04)	+ (1.78)	? (285.25)	? (300.00)
012_Template_Class	+ (0.01)	+ (0.17)	+ (6.56)	? (300.00)	? (274.53)
013_Cliques	+ (4.20)	+ (4.25)	+ (5.60)	? (300.00)	? (300.00)
014_Harry_belongs_to_some_Species	+ (0.00)	+ (0.05)	+ (1.52)	? (300.00)	? (300.00)
015_Reflective_Tautologies_I	+ (0.00)	+ (0.18)	+ (5.07)	? (300.95)	? (300.00)
016_Reflective_Tautologies_II	+ (0.03)	+ (0.16)	+ (3.17)	? (300.00)	? (300.00)
017_Builtin_Based_Definitions	+ (0.00)	+ (0.05)	+ (1.46)	? (300.00)	? (300.00)
018_Modified_Logical_Vocabulary_Semantics	+ (0.00)	+ (0.10)	+ (2.77)	? (300.00)	? (300.00)
019_Disjoint_Annotation_Properties	+ (0.00)	+ (0.03)	+ (1.45)	? (300.00)	? (300.00)
020_Logical_Complications	+ (31.08)	+ (31.06)	+ (34.26)	? (300.00)	? (300.00)
021_Composite_Enumerations	+ (3.79)	+ (3.86)	+ (5.21)	? (300.00)	? (300.00)
022_List_Member_Access	+ (0.02)	+ (0.20)	+ (7.69)	? (300.00)	? (300.00)
023_Unique_List_Components	+ (0.00)	+ (0.05)	+ (1.42)	? (300.00)	? (300.00)
024_Cardinality_Restrictions_on_Complex_Properties	+ (0.01)	+ (0.34)	+ (9.76)	? (300.00)	? (300.00)
025_Cyclic_Dependencies_between_Complex_Properties	+ (0.01)	+ (0.16)	+ (7.36)	? (300.00)	? (300.00)
026_Inferred_Property_Characteristics_I	+ (0.01)	+ (0.06)	+ (1.46)	? (300.00)	? (300.00)
027_Inferred_Property_Characteristics_II	+ (0.01)	+ (0.01)	+ (0.18)	+ (1.83)	? (300.00)
028_Inferred_Property_Characteristics_III	+ (0.02)	+ (0.03)	+ (0.18)	+ (1.81)	? (300.00)
029_Ex_Falso_Quodlibet	+ (0.00)	+ (0.04)	+ (1.52)	? (300.00)	? (300.00)
030_Bad_Class	+ (0.00)	+ (0.04)	+ (1.49)	? (300.00)	? (300.00)
031_Large_Universe	+ (0.01)	+ (0.13)	+ (3.04)	? (300.00)	? (300.00)
032_Datatype_Relationships	+ (0.00)	+ (0.05)	+ (1.49)	? (300.00)	? (300.00)

Table 15: Scalability results for the theorem prover *Vampire-SInE* using the small-sufficient OWL 2 Full subaxiomatizations.

Test Case	0	1200	10k	100k	1M
001_Subgraph_Entailment	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.80)	+ (19.53)
002_Existential_Blank_Nodes	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.76)	+ (20.15)
003_Blank_Nodes_for_Literals	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.81)	+ (19.52)
004_Axiomatic_Triples	+ (0.00)	+ (0.02)	+ (0.17)	+ (1.79)	+ (19.35)
005_Everything_is_a_Resource	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.78)	+ (19.54)
006_Literal_Values_represented_by_URIs_and_Blank_Nodes	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.80)	+ (19.54)
007_Equal_Classes	? (0.00)	? (0.01)	? (0.16)	? (1.79)	? (20.31)
008_Inverse_Functional_Data_Properties	+ (0.00)	+ (0.01)	+ (0.17)	+ (1.77)	+ (19.40)
009_Existential_Restriction_Entailments	+ (0.00)	+ (0.01)	+ (0.17)	+ (1.85)	+ (19.45)
010_Negative_Property_Assertions	+ (0.01)	+ (0.03)	+ (0.18)	+ (1.82)	+ (20.14)
011_Entity_Types_as_Classes	+ (0.00)	+ (0.01)	+ (0.17)	+ (1.76)	+ (19.43)
012_Template_Class	+ (0.20)	+ (0.23)	+ (0.38)	+ (2.05)	+ (20.40)
013_Cliques	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
014_Harry_belongs_to_some_Species	+ (0.32)	+ (0.34)	+ (0.50)	+ (2.15)	+ (19.84)
015_Reflective_Tautologies_I	+ (0.00)	+ (0.01)	+ (0.17)	+ (1.78)	+ (19.41)
016_Reflective_Tautologies_II	+ (0.00)	+ (0.02)	+ (0.17)	+ (1.79)	+ (19.52)
017_Builtin_Based_Definitions	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
018_Modified_Logical_Vocabulary_Semantics	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.84)	+ (19.44)
019_Disjoint_Annotation_Properties	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.79)	+ (19.45)
020_Logical_Complications	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
021_Composite_Enumerations	? (300.00)	? (300.00)	? (300.00)	? (300.00)	? (300.00)
022_List_Member_Access	+ (0.00)	+ (0.02)	+ (0.18)	+ (1.82)	+ (19.34)
023_Unique_List_Components	+ (0.00)	+ (0.02)	+ (0.17)	+ (1.78)	+ (19.77)
024_Cardinality_Restrictions_on_Complex_Properties	+ (0.00)	+ (0.02)	+ (0.17)	+ (1.80)	+ (19.53)
025_Cyclic_Dependencies_between_Complex_Properties	+ (0.01)	+ (0.03)	+ (0.18)	+ (1.79)	+ (19.43)
026_Inferred_Property_Characteristics_I	+ (0.00)	+ (0.02)	+ (0.17)	+ (1.80)	+ (19.42)
027_Inferred_Property_Characteristics_II	+ (0.03)	+ (0.05)	+ (0.19)	+ (1.81)	+ (19.36)
028_Inferred_Property_Characteristics_III	? (300.00)	+ (242.19)	+ (262.52)	? (300.00)	? (300.00)
029_Ex_Falso_Quodlibet	+ (0.00)	+ (0.02)	+ (0.17)	+ (1.78)	+ (19.47)
030_Bad_Class	+ (0.00)	+ (0.01)	+ (0.16)	+ (1.84)	+ (19.33)
031_Large_Universe	? (0.00)	? (0.01)	? (0.16)	? (1.80)	? (19.35)
032_Datatype_Relationships	+ (0.00)	+ (0.02)	+ (0.17)	+ (1.80)	+ (19.63)

A.3 Model Finding Results

The following table provides the raw result data that underlies the results for the model-finding experiments, as reported in Section 5.4. The only additional data here is the CPU time for each experiment. All experiments were conducted using the test suite of characteristic OWL 2 Full conclusions, as introduced in Section 4.2 (see Appendix B for more detailed information about the test suite).

Table 16: Model finding results for the model-finders *Paradox* and *DarwinFM* on the ALCO Full and RDFS axiom sets. The black entries indicate positive entailments or inconsistent ontologies. PA/A=Paradox/ALCO Full, PA/R=Paradox/RDFS, DF/R=DarwinFM/RDFS.

Test Case	PA/A	PA/R	DF/R
001_Subgraph_Entailment			
002_Existential_Blank_Nodes			
003_Blank_Nodes_for_Literals			
004_Axiomatic_Triples	+ (13.60)	+ (0.73)	+ (0.45)
005_Everything_is_a_Resource	+ (15.08)	+ (0.90)	+ (0.12)
006_Literal_Values_represented_by_URIs_and_Blank_Nodes	+ (20.95)	+ (0.81)	+ (0.04)
007_Equal_Classes	+ (13.01)	+ (1.03)	+ (7.19)
008_Inverse_Functional_Data_Properties	+ (11.74)	+ (0.99)	+ (0.08)
009_Existential_Restriction_Entailments		+ (1.17)	+ (0.05)
010_Negative_Property_Assertions		+ (1.61)	+ (0.07)
011_Entity_Types_as_Classes	+ (14.15)	+ (0.86)	+ (0.01)
012_Template_Class		+ (1.70)	+ (0.33)
013_Cliques	? (300.11)	+ (2.17)	+ (0.05)
014_Harry_belongs_to_some_Species		+ (1.16)	+ (0.56)
015_Reflective_Tautologies_I	+ (10.68)	+ (0.75)	+ (0.04)
016_Reflective_Tautologies_II	+ (8.21)	+ (0.77)	+ (2.05)
017_Builtin_Based_Definitions	+ (14.61)	+ (0.99)	+ (0.06)
018_Modified_Logical_Vocabulary_Semantics	+ (89.21)	+ (0.93)	+ (7.35)
019_Disjoint_Annotation_Properties	+ (14.55)	+ (0.89)	+ (0.01)
020_Logical_Complications	? (300.28)	+ (1.80)	+ (0.85)
021_Composite_Enumerations	? (300.15)	+ (2.21)	+ (0.11)
022_List_Member_Access	? (300.15)	+ (1.79)	+ (0.06)

023_Unique_List_Components	? (300.15)	+ (1.17)	+ (0.05)
024_Cardinality_Restrictions_on_Complex_Properties	+ (16.76)	+ (1.16)	+ (0.10)
025_Cyclic_Dependencies_between_Complex_Properties	? (300.17)	+ (1.65)	+ (0.06)
026_Inferred_Property_Characteristics_I	? (301.78)	+ (1.20)	+ (0.07)
027_Inferred_Property_Characteristics_II	? (300.12)	+ (1.04)	+ (0.07)
028_Inferred_Property_Characteristics_III	+ (17.62)	+ (1.10)	+ (0.07)
029_Ex_Falso_Quodlibet		+ (1.27)	+ (0.07)
030_Bad_Class	+ (17.88)	+ (1.05)	+ (0.01)
031_Large_Universe	? (300.55)	+ (0.93)	+ (0.01)
032_Datatype_Relationships	+ (9.69)	+ (0.85)	+ (0.06)

B OWL 2 Full Characteristic Conclusions Test Suite

This appendix presents the suite of OWL 2 Full-characteristic conclusion test cases that was used in the evaluation and has been introduced in Section 4.2. The appendix is divided into two parts: Section B.1 lists the *test cases*, and Section B.2 provides *correctness proofs* for them. The test suite is also available in electronic form as part of the *supplementary material* for this paper (see the download link at the beginning of Section 4), and can alternatively be obtained as a *separate package* from `http://www.fzi.de/downloads/ipe/schneid/testsuite-fullish.zip`.

B.1 Test Cases

Each test case is given by its *name*, its *type* (one of "Entailment" or "Inconsistency"), a textual *description*, and the *testing data* as one or two RDF graphs for an inconsistency test or entailment test, respectively. The RDF graphs are represented in *Turtle* syntax[5]. The electronic form of the test suite additionally contains serializations in *RDF/XML* syntax[6] and in the *TPTP* syntax [14].

001_Subgraph_Entailment (Entailment) In OWL 2 Full, a given RDF graph entails any of its sub graphs, even sub graphs that appear to encode broken language constructs of OWL. For example, the encoding of a class subsumption axiom that uses a property restriction as its superclass entails the single `owl:onProperty` triple of the serialization. This is a characteristic feature of the whole family of RDF-based languages, starting with RDF Simple Entailment, and it demonstrates the strictly triple-centered view that OWL 2 Full adopts. This behavior is typically shown by RDF entailment-rule reasoners, but not by OWL DL reasoners.

Premise Graph	**Conclusion Graph**

`ex:c rdfs:subClassOf ex:r .`
`ex:r rdf:type owl:Restriction .`
`ex:r owl:onProperty ex:p .`
`ex:r owl:someValuesFrom ex:d .`
	`ex:r rdf:type owl:Restriction .`
	`ex:r owl:onProperty ex:p .`

002_Existential_Blank_Nodes (Entailment) In OWL 2 Full, every blank node in an RDF graph is interpreted as an existentially quantified variable. On the one hand, this means that triples with URIs entail corresponding triples with blank nodes substituting the URIs. On the other hand, this means that triples with blank nodes entail corresponding triples with alternative blank nodes. This feature stems from RDF Simple Entailment. Many reasoners, in particular most RDF entailment-rule reasoners, do not provide the existential semantics of blank nodes.

[5] Turtle RDF syntax: `http://www.w3.org/TeamSubmission/turtle/`

[6] RDF/XML syntax: `http://www.w3.org/TR/rdf-syntax-grammar/`

Premise Graph	Conclusion Graph
ex:s ex:p _:o . _:o ex:q ex:s .	_:x ex:p _:y . _:y ex:q _:x .

003_Blank_Nodes_for_Literals (Entailment) In OWL 2 Full, an RDF triple having a data literal in object position entails a corresponding triple with a blank node substituting the literal. This feature stems from RDF Simple Entailment. It cannot be expected from OWL DL reasoners, since OWL 2 DL treats such blank nodes as anonymous individuals, while the domains of individuals and data values are defined to be disjoint. Most RDF entailment-rule reasoners do not show this behavior, since they typically do not implement the existential semantics of blank nodes.

Premise Graph	Conclusion Graph
ex:s ex:p "foo" .	ex:s ex:p _:x .

004_Axiomatic_Triples (Entailment) OWL 2 Full has many tautologies, i.e. statements that are entailed by the empty premise graph. Some of these tautologies have the form of "axiomatic triples", as defined by RDF and RDFS, but OWL 2 Full goes beyond these specifications. An example is the triple "`owl:Class rdfs:subClassOf owl:Thing`". RDF entailment-rule reasoners, such as OWL 2 RL/RDF rule reasoners, often prove at least some of the tautologies that OWL 2 Full provides, while for OWL 2 DL, many of these tautologies are not valid, neither syntactically nor semantically.

Premise Graph	Conclusion Graph
	owl:Class rdf:type owl:Thing . owl:Class rdf:type owl:Class . owl:Class rdfs:subClassOf owl:Thing . owl:Class owl:equivalentClass rdfs:Class . rdfs:Datatype rdfs:subClassOf owl:Class .

005_Everything_is_a_Resource (Entailment) In OWL 2 Full, following the semantics of RDFS, all three nodes of an RDF triple denote RDF resources (`rdfs:Resource`) and OWL individuals (`owl:Thing`). In addition, the predicate node of an RDF triple denotes an RDF property (`rdf:Property`) and an OWL object property (`owl:ObjectProperty`). RDF entailment-rule reasoners will often support this view to at least some extent. While OWL 2 DL offers some support for this view syntactically in the form of "punning", the strict separation of individuals, classes and properties in the semantics of OWL 2 DL prevents compliant OWL DL reasoners from producing many of the conclusions known from OWL 2 Full. In addition, OWL DL has only very limited support for RDF entity types such as `rdf:Property`.

Premise Graph	Conclusion Graph

`ex:s ex:p ex:o .`	`ex:s rdf:type rdfs:Resource .`
	`ex:s rdf:type owl:Thing .`
	`ex:p rdf:type rdfs:Resource .`
	`ex:p rdf:type owl:Thing .`
	`ex:p rdf:type rdf:Property .`
	`ex:p rdf:type owl:ObjectProperty .`
	`ex:o rdf:type rdfs:Resource .`
	`ex:o rdf:type owl:Thing .`

006_Literal_Values_represented_by_URIs_and_Blank_Nodes (Entailment) In OWL 2 Full, literals can be assigned URIs or blank nodes via `owl:sameAs` statements. One can then use these references to make further assertions about the literals and to draw semantic conclusions from them. This is an often discussed replacement for literals in the subject position of RDF triples, which is not supported by the RDF syntax. It is often supported by RDF entailment-rule reasoners to some extent, but is not allowed in OWL 2 DL, where URIs and blank nodes are used to refer to individuals but not to data values.

Premise Graph	Conclusion Graph

`ex:u owl:sameAs "abc" .`
`_:x owl:sameAs "abc" .`
`_:x owl:sameAs ex:w .`
	`ex:u owl:sameAs ex:w .`

007_Equal_Classes (Entailment) In OWL 2 Full, asserting that two classes are equal makes them into equivalent classes. This allows to substitute one class name for the other in all class-related axioms, such as class assertions, class subsumption axioms, and property range axioms. This can be observed in the Linked Open Data cloud, which contains many sameAs links between entities that are sometimes used as as classes in certain contexts. Many RDF entailment-rule reasoners provide for the expected semantic results. While syntactically allowed in OWL 2 DL via "punning", the semantic results are not available due to the strict separation of individuals and classes.

Premise Graph	Conclusion Graph

`ex:c1 owl:sameAs ex:c2 .`
`ex:w rdf:type ex:c1 .`
`ex:c rdfs:subClassOf ex:c1 .`
`ex:p rdfs:range ex:c1 .`
	`ex:w rdf:type ex:c2 .`
	`ex:c rdfs:subClassOf ex:c2 .`
	`ex:p rdfs:range ex:c2 .`

008_Inverse_Functional_Data_Properties (Entailment) In OWL 2 Full, data properties can be defined as inverse-functional properties. This option is, for example, frequently applied in the FOAF specification. While many RDF entailment-rule reasoners support the semantic consequences from these definitions, they are not supported by OWL 2 DL, which only allows object properties to be inverse-functional.

Premise Graph	Conclusion Graph
`foaf:mbox_sha1sum rdf:type owl:DatatypeProperty ; rdf:type owl:InverseFunctionalProperty . ex:bob foaf:mbox_sha1sum "xyz" . ex:robert foaf:mbox_sha1sum "xyz" .`	`ex:bob owl:sameAs ex:robert .`

009_Existential_Restriction_Entailments (Entailment) In OWL 2 Full, a class assertion using an existential property restriction entails a property assertion with a corresponding blank node. This inference is generally be provided by OWL DL reasoners, but in most cases is not provable by RDF entailment rule reasoners, which typically do not implement the existential semantics of blank nodes and existential property restrictions.

Premise Graph	Conclusion Graph
`ex:p rdf:type owl:ObjectProperty . ex:c rdf:type owl:Class . ex:s rdf:type [ rdf:type owl:Restriction ; owl:onProperty ex:p ; owl:someValuesFrom ex:c ] .`	`ex:s ex:p _:x . _:x rdf:type ex:c .`

010_Negative_Property_Assertions (Entailment) OWL 2 has introduced explicit support for negative property assertions (NPAs). However, it was already possible to encode NPAs in OWL 1, in terms of OWL 1 axioms and class expressions. These definitions are rather complex and require strong semantic support for several of the OWL language features. OWL 2 Full can infer that the new explicit encoding of NPAs follows from the corresponding old encoding of OWL 1. The same holds for OWL 2 DL. In contrast, RDF entailment-rule reasoners typically do not allow for such inferences due to the high semantic requirements.

Premise Graph	Conclusion Graph
`ex:p rdf:type owl:ObjectProperty . ex:s rdf:type [ owl:onProperty ex:p ; owl:allValuesFrom [ owl:complementOf [ owl:oneOf ( ex:o ) ] ] ] .`	`_:z rdf:type owl:NegativePropertyAssertion . _:z owl:sourceIndividual ex:s . _:z owl:assertionProperty ex:p . _:z owl:targetIndividual ex:o .`

011_Entity_Types_as_Classes (Inconsistency) In OWL 2 Full, entity types, such as `owl:Class`, are regular classes. This semantic property is basically inherited from RDFS. This makes it possible, for example, to state that the entity

types of classes and properties are mutually disjoint, and to infer inconsistencies if an entity is used as both a class and a property. Some RDF entailment rule reasoners, such as those implementing the OWL 2 RL/RDF rules, follow this semantics. OWL 2 DL, on the other hand, does not support it, since it sees entity types are purely syntactic information.

Graph

```
owl:Class owl:disjointWith owl:ObjectProperty .
ex:x rdf:type owl:Class .
ex:x rdf:type owl:ObjectProperty .
```

012_Template_Class (Entailment) In OWL 2 Full, instead of explicitly assigning features to a property, such as an entity type, property characteristics, or a domain, it is possible to build a class representing all these features and then make the property an instance of this "template class". Some RDF entailment rule reasoners, such as those implementing the OWL 2 RL/RDF rules, will support this approach to a certain extent, while in OWL 2 DL, in most cases it is be syntactically illegal and generally does not have the expected semantic meaning.

Premise Graph

```
foaf:Person rdf:type owl:Class .
ex:PersonAttribute owl:intersectionOf (
  owl:DatatypeProperty
  owl:FunctionalProperty [
    rdf:type owl:Restriction ;
    owl:onProperty rdfs:domain ;
    owl:hasValue foaf:Person
  ]
) .
ex:name rdf:type ex:PersonAttribute .
ex:alice ex:name "alice" .
```

Conclusion Graph

```
ex:name rdf:type owl:FunctionalProperty .
ex:alice rdf:type foaf:Person .
```

013_Cliques (Entailement) OWL 2 Full can define the metaclass of all cliques, for which each instance is a clique of people that know everyone else in that clique. The encoding is not supported by OWL 2 DL, since it uses built-in vocabulary terms as regular entities. For RDF entailment rule reasoners, the semantic requirements for producing all expected results are typically too high.

Premise Graph	Conclusion Graph

ex:Clique rdf:type owl:Class .
ex:sameCliqueAs
  rdfs:subPropertyOf owl:sameAs ;
  rdfs:range ex:Clique .
ex:Clique rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty ex:sameCliqueAs ;
  owl:someValuesFrom ex:Clique
] .
foaf:knows
  rdf:type owl:ObjectProperty ;
  owl:propertyChainAxiom (
  rdf:type
  ex:sameCliqueAs
  [owl:inverseOf rdf:type]
) .
ex:JoesGang rdf:type ex:Clique .
ex:alice rdf:type ex:JoesGang .
ex:bob rdf:type ex:JoesGang .
	ex:alice foaf:knows ex:bob .

014_Harry_belongs_to_some_Species (Entailment) OWL 2 Full supports the combination of metamodelling and class union. For example, provided that the classes of eagles and falcons are both instances of the metaclass of species, if one does not exactly know whether Harry is an eagle or a falcon, one can still conclude that Harry must belong to some species. OWL 2 DL does not support semantic conclusions from metamodeling, although it allows for some metamodeling syntactically via "punning". While many RDF entailment-rule reasoners have some restricted support for semantic metamodeling, drawing said conclusion from the union of classes typically goes beyond the capabilities of these reasoners.

Premise Graph	Conclusion Graph
ex:Eagle rdf:type ex:Species . ex:Falcon rdf:type ex:Species . ex:harry rdf:type [owl:unionOf (ex:Eagle ex:Falcon)] .	ex:harry rdf:type _:x . _:x rdf:type ex:Species .

015_Reflective_Tautologies_I (Entailment) In OWL 2 Full, the statement "`owl:sameAs owl:sameAs owl:sameAs`" is a tautology. This is a classic example used to demonstrate the use of built-in vocabulary terms as regular entities, sometimes referred to as "syntax reflection". It is not allowed in OWL 2 DL. Some RDF entailment-rule reasoners, such as those implementing the OWL 2 RL/RDF rules, do provide this result.

Premise Graph	Conclusion Graph
	owl:sameAs owl:sameAs owl:sameAs .

016_Reflective_Tautologies_II (Entailment) In OWL 2 Full, the class equivalence property is a subproperty of the class subsumption property. This is an example of the use of built-in vocabulary terms as regular entities, occasionally referred to as "syntax reflection". It is not allowed in OWL 2 DL. RDF entailment-rule reasoners may contain this tautology as a special rule, but otherwise cannot be expected to provide this result. For example, the result does not follow from the OWL 2 RL/RDF rules.

Premise Graph	**Conclusion Graph**
	`owl:equivalentClass` `  rdfs:subPropertyOf rdfs:subClassOf .`

017_Builtin_Based_Definitions (Entailment) In OWL 2 Full, custom properties can be defined based on existing built-in properties. For example, a property `ex:noInstanceOf` that is disjoint from `rdf:type` can be defined, and this new property can be used to state non-membership, which has semantic ramifications. OWL 2 DL does not allow this. Entailment-rule reasoners can make such assertions, and may provide some limited support for semantic conclusions.

Premise Graph	**Conclusion Graph**

`ex:notInstanceOf`
`  owl:propertyDisjointWith rdf:type .`
`ex:w rdf:type ex:c .`
`ex:u ex:notInstanceOf ex:c .`
	`ex:w owl:differentFrom ex:u .`

018_Modified_Logical_Vocabulary_Semantics (Entailment) The semantics of OWL built-in vocabulary terms can be enriched in a way such that their application leads to additional results that are not available from their original meaning. For example, the domain and range of `owl:sameAs` can be restricted to the class of persons, which renders all things that are equal into persons. OWL 2 DL does not allow this, while RDF entailment-rule reasoners often provide some limited support.

Premise Graph	**Conclusion Graph**
`owl:sameAs rdfs:domain ex:Person .` `ex:w owl:sameAs ex:u .`	`ex:u rdf:type ex:Person .`

019_Disjoint_Annotation_Properties (Inconsistency) In OWL 2 Full, annotation properties are normal object properties. Thus, two annotation properties can be specified to be disjoint, and semantic conclusions can be drawn from this disjointness. This feature is, for example, used in the SKOS specification to define the meaning of lexical labels. OWL 2 DL provides only limited

syntactic support for putting axioms on annotation properties, and does not provide any semantic conclusions. One can expect limited semantic support from some RDF entailment-rule reasoners, such as those implementing the OWL 2 RL/RDF rules.

Graph

```
skos:prefLabel rdf:type owl:AnnotationProperty .
skos:prefLabel rdfs:subPropertyOf rdfs:label .
skos:altLabel rdf:type owl:AnnotationProperty .
skos:altLabel rdfs:subPropertyOf rdfs:label .
skos:prefLabel owl:propertyDisjointWith skos:altLabel .
ex:foo skos:prefLabel "foo" .
ex:foo skos:altLabel "foo" .
```

020_Logical_Complications (Entailment) OWL 2 Full allows complex logical reasoning to be performed. For example, non-obvious subsumption relationships between two classes can be inferred based on the application of disjointness and different Boolean connectives. This kind of reasoning is generally possible in unrestricted form in OWL 2 DL, but typically not with RDF entailment-rule reasoners.

Premise Graph

```
ex:c owl:unionOf ( ex:c1 ex:c2 ex:c3 ) .
ex:d owl:disjointWith ex:c1 .
ex:d rdfs:subClassOf [
  owl:intersectionOf (
    ex:c
    [ owl:complementOf ex:c2 ]
  )
] .
```

Conclusion Graph

```
ex:d rdfs:subClassOf ex:c3 .
```

021_Composite_Enumerations (Entailment) OWL 2 Full allows for the composition of enumerations via boolean connectives. For example, the union of the classes $\{w1, w2\}$ and $\{w2, w3\}$ can be inferred to be equivalent to the class $\{w1, w2, w3\}$. OWL 2 DL reasoners can be expected to provide this result, while RDF entailment-rule reasoners are typically unable to produce the result.

Premise Graph

```
ex:c1 owl:oneOf ( ex:w1 ex:w2 ) .
ex:c2 owl:oneOf ( ex:w2 ex:w3 ) .
ex:c3 owl:oneOf ( ex:w1 ex:w2 ex:w3 ) .
ex:c4 owl:unionOf ( ex:c1 ex:c2 ) .
```

Conclusion Graph

```
ex:c3 owl:equivalentClass ex:c4 .
```

022_List_Member_Access (Entailment) In OWL 2 Full, one can refer to all items within an RDF list. For example, Chapter 9 of the SKOS Reference defines ordered concept collections via the property `skos:memberList` applied to some RDF list consisting of items of type `skos:Concept`. SKOS further defines

non-ordered concept collections by applying the property `skos:member` repeatedly to single entities of type `skos:Concept`. SKOS statement S36 says that a non-ordered concept collection can be inferred from an ordered collection. An example is given in Section 9.6.1 of the SKOS Reference. OWL 2 Full allows this statement to be expressed semantically. Both the encoding of S36 and the example inference is given here. RDF entailment-rule reasoners implementing the OWL 2 RL/RDF rules also produce the result. OWL 2 DL cannot make assertions about RDF lists.

Premise Graph	**Conclusion Graph**

`skos:memberList rdfs:subPropertyOf _:pL .`
`skos:member owl:propertyChainAxiom (`
`  _:pL`
`  rdf:first`
`) .`
`_:pL owl:propertyChainAxiom (`
`  _:pL`
`  rdf:rest`
`) .`
`ex:MyOrderedCollection`
`  rdf:type skos:OrderedCollection ;`
`  skos:memberList ( ex:X ex:Y ex:Z ) .`
	`ex:MyOrderedCollection skos:member ex:X .`
	`ex:MyOrderedCollection skos:member ex:Y .`
	`ex:MyOrderedCollection skos:member ex:Z .`

023_Unique_List_Components (Entailment) In principle, it is possible to create argument lists of OWL constructs that are non-linear. Section 3.3.3 of the RDF Semantics specification allows semantic extensions to place extra syntactic wellformedness restrictions on the use of the RDF Collections vocabulary in order to rule out graphs containing non-linear lists. While OWL 2 Full does not provide this directly, it can state that the List vocabulary property `rdf:first` is a functional property. This has semantic consequences even if the argument list of an OWL construct is given in a non-linear form. RDF entailment-rule reasoners often have some limited support for these kinds of results. OWL 2 DL cannot make assertions about RDF lists.

Premise Graph	**Conclusion Graph**

`rdf:first rdf:type owl:FunctionalProperty .`
`ex:w rdf:type [`
`  rdf:type owl:Class ;`
`  owl:oneOf _:l`
`] .`
`_:l rdf:first ex:u .`
`_:l rdf:first ex:v .`
`_:l rdf:rest rdf:nil .`
	`ex:w owl:sameAs ex:u .`
	`ex:w owl:sameAs ex:v .`

024_Cardinality_Restrictions_on_Complex_Properties (Entailment) OWL 2 DL does cannot place cardinality restrictions on transitive properties. OWL 2 Full allows this. This can, for example, be used to state that every person has at least one ancestor. The existence of an ancestor can then be inferred for any given person. RDF entailment-rule reasoners may provide some limited support but typically are unable to produce the result of this particular example.

Premise Graph	Conclusion Graph

ex:hasAncestor
  rdf:type owl:TransitiveProperty .
ex:Person rdfs:subClassOf [
  rdf:type owl:Restriction ;
  owl:onProperty ex:hasAncestor ;
  owl:minCardinality
    "1"^^xsd:nonNegativeInteger
] .
ex:alice rdf:type ex:Person .
ex:bob rdf:type ex:Person .
ex:alice ex:hasAncestor ex:bob .
	ex:bob ex:hasAncestor _:x .
	ex:alice ex:hasAncestor _:x .

025_Cyclic_Dependencies_between_Complex_Properties (Entailment) OWL 2 DL does not allow cyclic dependencies between complex properties that are defined via subproperty chain axioms. OWL 2 Full allows this. For example, the uncle relation and the cousin relation can be expressed mutually in terms of the other relation using two subproperty chain axioms. This provides for more precise characterizations of properties than it is possible in OWL 2 DL. RDF entailment rule reasoners that implement the OWL 2 RL/RDF rules provide limited support for reasoning in such scenarios.

Premise Graph	Conclusion Graph

ex:hasUncle owl:propertyChainAxiom (
  ex:hasCousin
  ex:hasFather
) .
ex:hasCousin owl:propertyChainAxiom (
  ex:hasUncle
  [owl:inverseOf ex:hasFather]
) .
ex:alice ex:hasFather ex:dave .
ex:alice ex:hasCousin ex:bob .
ex:bob ex:hasFather ex:charly .
ex:bob ex:hasUncle ex:dave .
	ex:alice ex:hasUncle ex:charly .
	ex:bob ex:hasCousin ex:alice .

026_Inferred_Property_Characteristics_I (Entailment) In OWL 2 Full, as in OWL 2 DL, a property that has a domain and a range being singleton classes is entailed to be an inverse-functional property. RDF entailment-rule reasoners cannot be expected to provide this result, since it requires sophisticated reasoning.

Premise Graph	Conclusion Graph
ex:p rdfs:domain [owl:oneOf (ex:w)] . ex:p rdfs:range [owl:oneOf (ex:u)] .	ex:p rdf:type owl:InverseFunctionalProperty .

027_Inferred_Property_Characteristics_II (Entailment) In OWL 2 Full, if the chain of a property and its inverse property builds a subproperty chain of `owl:sameAs`, then that property is inverse-functional. The application of the

built-in vocabulary term `owl:sameAs` is not allowed in OWL 2 DL. Newer RDF entailment-rule reasoners, such as those implementing the OWL 2 RL/RDF rules, may provide some limited semantic support.

Premise Graph	Conclusion Graph

`owl:sameAs owl:propertyChainAxiom (`
`  ex:p`
`  [owl:inverseOf ex:p]`
`) .`
	`ex:p rdf:type owl:InverseFunctionalProperty .`

028_Inferred_Property_Characteristics_III (Entailment) In OWL 2 Full, instead of using the built-in property characteristics of inverse-functional properties, properties can be made into instances of the custom class of the inverses of all functional properties. OWL 2 DL does not allow the use of built-in vocabulary terms as regular entities. For RDF entailment-rule reasoners, the semantic result given in this example is typically too demanding.

Premise Graph	Conclusion Graph

`ex:InversesOfFunctionalProperties`
`  owl:equivalentClass [`
`    rdf:type owl:Restriction ;`
`    owl:onProperty owl:inverseOf ;`
`    owl:someValuesFrom owl:FunctionalProperty`
`  ] .`
	`ex:InversesOfFunctionalProperties`
	`  rdfs:subClassOf owl:InverseFunctionalProperty .`

029_Ex_Falso_Quodlibet (Entailment) In OWL 2 Full, an inconsistent premise ontology entails arbitrary conclusion ontologies ("principle of explosion", "ex falso sequitur quodlibet"). OWL 2 DL has the same semantic property, but many existing OWL 2 DL reasoners signal an error when given an inconsistent premise ontology, and do not produce the expected result (however, it would be trivial to extend an OWL 2 DL reasoner to give the result as a reaction to an inconsistency error). RDF entailment-rule reasoners cannot be expected to produce tyhis result, since it requires full semantic support for classical negation.

Premise Graph	Conclusion Graph

`ex:A rdf:type owl:Class .`
`ex:B rdf:type owl:Class .`
`ex:w rdf:type [`
`  owl:intersectionOf (`
`    ex:A`
`    [owl:complementOf ex:A]`
`  )`
`] .`
	`ex:w rdf:type ex:B .`

030_Bad_Class (Inconsistency) If an OWL 2 Full ontology contains a class that has the Russell Set as its class extension, then the ontology is inconsistent.

This situation would occur for even the empty ontology if the so-called OWL 2 Full comprehension conditions, as non-normatively defined in Chapter 8 of the OWL 2 RDF-Based Semantics, were a normative part of OWL 2 Full, as explained in Chapter 9 of the specification document. OWL 2 DL does not know about this issue, and RDF entailment-rule reasoners cannot be expected to know about it due to their relatively weak semantics.

Graph

```
ex:c rdf:type owl:Class .
ex:c owl:complementOf [
  rdf:type owl:Restriction ;
  owl:onProperty rdf:type ;
  owl:hasSelf "true"^^xsd:boolean
] .
```

031_Large_Universe (Inconsistency) The universe of an OWL 2 Full interpretation cannot consist of only a single individual. This means that `owl:Thing` cannot be equivalent to a singleton enumeration class, without leading to an inconsistent ontology. This is different from OWL 2 DL, for which the only restriction on the universe is that it has to be non-empty. RDF entailment-rule reasoners cannot be expected to provide the inconsistency result, since this requires strong logic-based reasoning.

Graph

```
owl:Thing owl:equivalentClass [
  owl:oneOf ( ex:w )
] .
```

032_Datatype_Relationships (Entailment) According to the XSD Datatypes specification, the value spaces of the datatypes `xsd:decimal` and `xsd:string` are disjoint, while the value space of `xsd:integer` is a subset of the value space of `xsd:decimal`. In OWL 2 Full, these relationships between the data values of datatypes can be observed as corresponding relationships between the classes representing these datatypes. OWL 2 DL also follows the XSD semantics, but it does not support to explicitly query for subsumption or disjointness relationships between datatypes. Some RDF entailment-rule reasoners may decide to provide the different relationships between XSD datatypes as explicit facts or rules, but cannot, in general, be expected to do so.

Premise Graph	Conclusion Graph
	`xsd:decimal owl:disjointWith xsd:string .` `xsd:integer rdfs:subClassOf xsd:decimal .`

B.2 Correctness Proofs

This section provides *correctness proofs* for all test cases listed in Section B.1. The proofs have been constructed with respect to the OWL 2 RDF Based Semantics [12] and the RDF Semantics [5], which conjointly specify the model-theoretic semantics of OWL 2 Full.

001_Subgraph_Entailment (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph, so the following becomes true:

$$\begin{array}{l}\langle I(\texttt{ex:c}), I(\texttt{ex:r})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\\ \langle I(\texttt{ex:r}), I(\texttt{owl:Restriction})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\\ \langle I(\texttt{ex:r}), I(\texttt{ex:p})\rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\\ \langle I(\texttt{ex:r}), I(\texttt{ex:d})\rangle \in \mathrm{IEXT}(I(\texttt{owl:someValuesFrom}))\end{array}$$

Then, in particular, the conjunction of the subset of atoms

$$\begin{array}{l}\langle I(\texttt{ex:r}), I(\texttt{owl:Restriction})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\\ \langle I(\texttt{ex:r}), I(\texttt{ex:p})\rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\end{array}$$

is also satisfied.

002_Existential_Blank_Nodes (Proof) Let I be an OWL 2 RDF-Based interpretation interpretation and B be a blank node mapping for the blank nodes in the premise graph, such that $I + B$ satisfies the premise graph. This gives

$$\exists o : \langle I(\texttt{ex:s}), o\rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \ \wedge\ \langle o, I(\texttt{ex:s})\rangle \in \mathrm{IEXT}(I(\texttt{ex:q}))$$

Weakening this statement by introducing an existentially quantified variable for $I(\texttt{ex:s})$ logically implies

$$\exists x, y : \langle x, y\rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \ \wedge\ \langle y, x\rangle \in \mathrm{IEXT}(I(\texttt{ex:q}))$$

Thus, there is a blank node mapping B', such that $I + B'$ satisfies the conclusion graph.

003_Blank_Nodes_for_Literals (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph. Then from

$$\langle I(\texttt{ex:s}), I(\texttt{"foo"})\rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))$$

and taking into account that literals denote individuals in the universe, we receive the formally weaker assertion

$$\exists x : \langle I(\texttt{ex:s}), x\rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))$$

Thus, there is a blank node mapping B, such that $I + B$ satisfies the conclusion graph.

004_Axiomatic_Triples (Proof) Given a satisfying OWL 2 RDF-Based interpretation I for the empty graph.
1) Claim: $\langle I(\texttt{owl:Class}), I(\texttt{owl:Thing})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))$.
Proof: The denotation of `owl:Class` is in the universe, i.e., `I(owl:Class)` $\in$ IR. The claim follows from $\mathrm{ICEXT}(I(\texttt{owl:Thing})) = \mathrm{IR}$ (OWL2/Tab5.2) and from the RDFS semantic condition defining "ICEXT".
2) Claim: $\langle I(\texttt{owl:Class}), I(\texttt{owl:Class})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))$.
Proof: $I(\texttt{owl:Class}) \in \mathrm{IC}$ and $\mathrm{ICEXT}(I(\texttt{owl:Class})) = \mathrm{IC}$ (OWL2/Tab5.2). The claim follows from the RDFS semantic condition defining "ICEXT".
3) Claim: $\langle I(\texttt{owl:Class}), I(\texttt{owl:Thing})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))$.
Proof: According to 2), $I(\texttt{owl:Class}) \in \mathrm{IC}$. Further, $I(\texttt{owl:Thing}) \in \mathrm{IC}$ according to OWL2/Tab5.2. Given arbitrary $x \in \mathrm{ICEXT}(I(\texttt{owl:Class}))$, then $x \in \mathrm{IR}$, and thus $x \in \mathrm{ICEXT}(I(\texttt{owl:Thing}))$ according to OWL2/Tab5.2. The claim follows from using the "$\leftarrow$" direction of the OWL 2 semantic condition for class subsumption (OWL2/Tab5.8).
4) Claim: $\langle I(\texttt{owl:Class}), I(\texttt{rdfs:Class})\rangle \in \mathrm{IEXT}(I(\texttt{owl:equivalentClass}))$.
Proof: According to 2), we get $I(\texttt{owl:Class}) \in \mathrm{IC}$. According to OWL2/Tab5.2, we get $I(\texttt{rdfs:Class}) \in \mathrm{IC}$. From OWL2/Tab5.2 we get $\mathrm{ICEXT}(I(\texttt{owl:Class})) = \mathrm{IC} = \mathrm{ICEXT}(I(\texttt{rdfs:Class}))$. The claim follows from using the "$\leftarrow$" direction of the OWL 2 semantic condition for class equivalence (OWL2/Tab5.9).
5) Claim: $\langle I(\texttt{rdfs:Datatype}), I(\texttt{owl:Class})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))$.
Proof: According to 2), we get $I(\texttt{owl:Class}) \in \mathrm{IC}$. According to OWL2/Tab5.2, we get $I(\texttt{rdfs:Datatype}) \in \mathrm{IC}$. Given arbitrary $x \in \mathrm{ICEXT}(I(\texttt{rdfs:Datatype}))$. By OWL2/Tab5.2 we get $x \in \mathrm{IDC}$. Then, OWL2/Tab5.1 gives $x \in \mathrm{IC}$. Finally, OWL2/Tab5.2 gives $x \in \mathrm{ICEXT}(I(\texttt{owl:Class}))$. The claim now follows from the "$\leftarrow$" direction of the OWL 2 semantic condition for class subsumption (OWL2/Tab5.8).

005_Everything_is_a_Resource (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph.
1a) Claim: $I(\texttt{ex:s}) \in \mathrm{IR}$, $I(\texttt{ex:p}) \in \mathrm{IR}$, $I(\texttt{ex:o}) \in \mathrm{IR}$.
Proof: Since I is a simple-interpretation, $I(\texttt{ex:s})$ and $I(\texttt{ex:o})$ are in IR, and $I(\texttt{ex:p})$ is in IP. According to the RDF semantic condition that defines "IP", $\langle I(\texttt{ex:p}), I(\texttt{rdf:Property})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type})$, and thus $I(\texttt{ex:p}) \in \mathrm{IR}$.
1b) Claim: $\langle I(\texttt{ex:s}), I(\texttt{rdfs:Resource})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))$, and ditto for `ex:p` and `ex:o`.
Proof: 1a) showed $I(\texttt{ex:s}) \in IR$, and from the RDFS semantic condition that defines the class extension of `rdfs:Resource` to be the set IR follows $I(\texttt{ex:s}) \in \mathrm{ICEXT}(I(\texttt{rdfs:Resource}))$. The claim follows from the RDFS semantic condition that defines "ICEXT". Analog proofs apply to `ex:p` and `ex:o`.
1c) Claim: $\langle I(\texttt{ex:s}), I(\texttt{owl:Thing})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))$, and ditto for `ex:p` and `ex:o`.
Proof: As for 1b), but applying the OWL 2 semantic condition that defines the extension of `owl:Thing` (OWL2/Tab5.2) instead of `rdfs:Resource`.
2a) Claim: $I(\texttt{ex:p}) \in \mathrm{IP}$.

Proof: This follows directly from I being a simple-interpretation that satisfies the premise graph.
2b) Claim: $\langle I(\texttt{ex:p}), I(\texttt{rdf:Property})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$.
Proof: According to 2a), $I(\texttt{ex:p}) \in \text{IP}$, and the RDF semantic condition that defines "IP" provides the claim.
2c) Claim: $\langle I(\texttt{ex:p}), I(\texttt{owl:ObjectProperty})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$.
Proof: According to 2a), $I(\texttt{ex:p}) \in \text{IP}$, and according to OWL2/Tab5.2 the class extension of $I(\texttt{owl:ObjectProperty})$ is IP. The claim follows from the RDFS semantic extension that defines "ICEXT".

006_Literal_Values_represented_by_URIs_and_Blank_Nodes (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I+B$ satisfies the premise graph. Given an x, such that

$$\begin{aligned}&(1)\ \langle I(\texttt{ex:u}), I(\texttt{"abc"})\rangle \in \text{IEXT}(I(\texttt{owl:sameAs})), \text{ and}\\&(2)\ \langle x, I(\texttt{"abc"})\rangle \in \text{IEXT}(I(\texttt{owl:sameAs})), \text{ and}\\&(3)\ \langle x, I(\texttt{ex:w})\rangle \in \text{IEXT}(I(\texttt{owl:sameAs})).\end{aligned}$$

By the "→" direction of the semantic condition for `owl:sameAs` (OWL2/Tab5.9), we receive that

$$\begin{aligned}&(1')\ I(\texttt{ex:u}) = I(\texttt{"abc"}), \text{ and}\\&(2')\ x = I(\texttt{"abc"}), \text{ and}\\&(3')\ x = I(\texttt{ex:w}).\end{aligned}$$

From (2') and (3') we conclude that

$$(4)\ I(\texttt{"abc"}) = I(\texttt{ex:w}).$$

From (1') and (4) we conclude

$$(5)\ I(\texttt{ex:u}) = I(\texttt{ex:w}).$$

From the "←" direction of the semantic condition for `owl:sameAs` (OWL2/Tab5.9), we conclude

$$(6)\langle I(\texttt{ex:u}), I(\texttt{ex:w})\rangle \in \text{IEXT}(I(\texttt{owl:sameAs})).$$

007_Equal_Classes (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph. From the fact

$$\langle I(\texttt{ex:c1}), I(\texttt{ex:c2})\rangle \in \text{IEXT}(I(\texttt{owl:sameAs}))$$

the "→" direction of the semantic condition for `owl:sameAs` (OWL2/Tab5.9) provides

$$I(\texttt{ex:c1}) = I(\texttt{ex:c2}).$$

So we can substitute any occurrence of $I(\texttt{ex:c1})$ by $I(\texttt{ex:c2})$. Hence, from the premises

$$\begin{aligned}
&\langle I(\texttt{ex:w}), I(\texttt{ex:c1})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\\
&\langle I(\texttt{ex:c}), I(\texttt{ex:c1})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\\
&\langle I(\texttt{ex:p}), I(\texttt{ex:c1})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:range}))
\end{aligned}$$

we receive the corresponding conclusions

$$\begin{aligned}
&\langle I(\texttt{ex:w}), I(\texttt{ex:c2})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\\
&\langle I(\texttt{ex:c}), I(\texttt{ex:c2})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\\
&\langle I(\texttt{ex:p}), I(\texttt{ex:c2})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:range}))
\end{aligned}$$

008_Inverse_Functional_Data_Properties (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph. We start from

$$\begin{aligned}
&(1a)\ \langle I(\texttt{ex:bob}), I(\texttt{"xyz"})\rangle \in \mathrm{IEXT}(I(\texttt{foaf:mbox_sha1sum})), \text{ and}\\
&(1b)\ \langle I(\texttt{ex:robert}), I(\texttt{"xyz"})\rangle \in \mathrm{IEXT}(I(\texttt{foaf:mbox_sha1sum})),
\end{aligned}$$

as well as from

$$(2)\ \langle I(\texttt{foaf:mbox\_sha1sum}), I(\texttt{owl:InverseFunctionalProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type})).$$

From (2) and the "→" direction of the RDFS semantic condition for ICEXT we receive

$$(2')\ I(\texttt{foaf:mbox\_sha1sum}) \in \mathrm{ICEXT}(I(\texttt{owl:InverseFunctionalProperty})).$$

This allows to apply the "→" direction of semantic condition for inverse-functional properties (OWL2/Tab5.13), which provides

$$\begin{aligned}
(3)\ &\forall x_1, x_2, y:\\
&\quad \langle x_1, y\rangle \in \mathrm{IEXT}(I(\texttt{foaf:mbox_sha1sum})) \wedge\\
&\quad \langle x_2, y\rangle \in \mathrm{IEXT}(I(\texttt{foaf:mbox_sha1sum}))\\
&\quad \Rightarrow x_1 = x_2
\end{aligned}$$

Applying (3) to (1a) and (1b) with

$$\begin{aligned}
x_1 &:= I(\texttt{ex:bob}),\\
x_2 &:= I(\texttt{ex:robert}), \text{ and}\\
y &:= I(\texttt{"xyz"})
\end{aligned}$$

results in

$$(4)\ I(\texttt{ex:bob}) = I(\texttt{ex:robert}).$$

Using the "←" direction of the semantic condition for `owl:sameAs` (OWL2/Tab5.9) results in

$$(5)\ \langle I(\texttt{ex:bob}), I(\texttt{ex:robert})\rangle \in \mathrm{IEXT}(I(\texttt{owl:sameAs})).$$

009_Existential_Restriction_Entailments (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Given a z such that the following holds:

$$
\begin{aligned}
&(1a)\ \langle I(\texttt{ex:p}), I(\texttt{owl:ObjectProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1b)\ \langle I(\texttt{ex:c}), I(\texttt{owl:Class})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1c)\ \langle I(\texttt{ex:s}), z\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1d)\ \langle z, I(\texttt{owl:Restriction})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1e)\ \langle z, I(\texttt{ex:p})\rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\ ,\\
&(1f)\ \langle z, I(\texttt{ex:c})\rangle \in \mathrm{IEXT}(I(\texttt{owl:someValuesFrom}))\ .
\end{aligned}
$$

From the RDFS semantic condition for ICEXT ("→" direction) and (1c) follows (2) $I(\texttt{ex:s}) \in \mathrm{ICEXT}(z)$. From the semantic condition of `owl:someValuesFrom` (OWL2/Tab5.6), (1e) and (1f) follows

$$
\begin{aligned}
&(3)\ \forall y : y \in \mathrm{ICEXT}(z) \Leftrightarrow\\
&\qquad \exists x :\ [\,\langle y, x\rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))\ \wedge\ x \in \mathrm{ICEXT}(I(\texttt{ex:c}))\,]\ .
\end{aligned}
$$

From (2) and (3) follows

$$(4)\ \exists x :\ \langle I(\texttt{ex:s}), x\rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))\ \wedge\ x \in \mathrm{ICEXT}(I(\texttt{ex:c}))\ .$$

By the RDFS semantic condition for ICEXT ("←" direction) and (4) we receive

$$(5)\ \exists x :\ \langle I(\texttt{ex:s}), x\rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))\ \wedge\ \langle x, I(\texttt{ex:c})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

010_Negative_Property_Assertions (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let x_1, x_2, x_3 and x_4 be individuals, such that the following holds

$$
\begin{aligned}
&(1a)\ \langle I(\texttt{ex:p}), I(\texttt{owl:ObjectProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1b)\ \langle I(\texttt{ex:s}), x_1\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1c)\ \langle x_1, I(\texttt{ex:p})\rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\ ,\\
&(1d)\ \langle x_1, x_2\rangle \in \mathrm{IEXT}(I(\texttt{owl:allValuesFrom}))\ ,\\
&(1e)\ \langle x_2, x_3\rangle \in \mathrm{IEXT}(I(\texttt{owl:complementOf}))\ ,\\
&(1f)\ \langle x_3, x_4\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
&(1g)\ \langle x_4, I(\texttt{ex:o})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1h)\ \langle x_4, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ .
\end{aligned}
$$

From the RDFS semantic condition for ICEXT ("→" direction) and (1b) follows

$$(2b)\ I(\texttt{ex:s}) \in \mathrm{ICEXT}(x_1)\ .$$

From the semantic condition for `owl:allValuesFrom` (OWL2/Tab5.6), (1c) and (1d) follows

$$
\begin{aligned}
&(2c)\ \forall y :\ y \in \mathrm{ICEXT}(x_1) \Leftrightarrow\\
&\qquad \forall z :\ [\langle y, z\rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \Rightarrow z \in \mathrm{ICEXT}(x_2)]\ .
\end{aligned}
$$

From the "→" direction of the semantic condition for class complement (OWL2/Tab5.4) and (1e) follows

$$(2e)\ \forall y:\ y \in \mathrm{ICEXT}(x_2) \Leftrightarrow y \notin \mathrm{ICEXT}(x_3)\ .$$

From the "→" direction of the semantic condition for singleton enumerations (OWL2/Tab5.5) and (1f), (1g) and (1h) follows

$$(2f)\ \forall y:\ y \in \mathrm{ICEXT}(x_3) \Leftrightarrow y = I(\texttt{ex:o})\ .$$

Assume that

$$(3)\ \langle I(\texttt{ex:s}), I(\texttt{ex:o})\rangle \in IEXT(I(\texttt{ex:p}))\ .$$

By (2b), (2c) and (3) we receive

$$(4)\ I(\texttt{ex:o}) \in \mathrm{ICEXT}(x_2)\ .$$

By (4) and (2e) follows

$$(5)\ I(\texttt{ex:o}) \notin \mathrm{ICEXT}(x_3)\ .$$

By (5) and (2f) follows

$$(6)\ I(\texttt{ex:o}) \neq I(\texttt{ex:o})\ .$$

This is a contradiction, hence assumption (3) was wrong. So we have:

$$(3')\ \langle I(\texttt{ex:s}), I(\texttt{ex:o})\rangle \notin \mathrm{IEXT}(I(\texttt{ex:p}))\ .$$

Since I is a simple-interpretation, we receive

$$(7a)\ I(\texttt{ex:s}) \in \mathrm{IR}\ ,$$
$$(7b)\ I(\texttt{ex:o}) \in \mathrm{IR}\ .$$

By the property extension of `owl:onProperty` (OWL2/Tab5.3) follows

$$(7c)\ I(\texttt{ex:p}) \in \mathrm{IP}\ .$$

From the second semantic condition for NPAs in OWL2/Tab5.15, (3'), (7a), (7b) and (7c) follows that there exists some z, such that

$$(8a)\ \langle z, I(\texttt{ex:s})\rangle \in \mathrm{IEXT}(I(\texttt{owl:sourceIndividual}))\ ,$$
$$(8b)\ \langle z, I(\texttt{ex:p})\rangle \in \mathrm{IEXT}(I(\texttt{owl:assertionProperty}))\ ,$$
$$(8c)\ \langle z, I(\texttt{ex:o})\rangle \in \mathrm{IEXT}(I(\texttt{owl:targetIndividual}))\ .$$

Finally, from the property extension of `owl:sourceIndividual` and (8a) follows

$$(9)\ z \in \mathrm{ICEXT}(I(\texttt{owl:NegativePropertyAssertion}))\ ,$$

which by the "←" direction of the RDFS semantic condition for ICEXT results in

$$(9')\ \langle z, I(\texttt{owl:NegativePropertyAssertion})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

The result consists of (9), (8a), (8b) and (8c).

011_Entity_Types_as_Classes (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph. We start from the facts:

$(1a)$ $\langle I(\texttt{owl:Class}), I(\texttt{owl:ObjectProperty})\rangle \in \text{IEXT}(I(\texttt{owl:disjointWith}))$,
$(1b)$ $\langle I(\texttt{ex:x}), I(\texttt{owl:Class})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$,
$(1c)$ $\langle I(\texttt{ex:x}), I(\texttt{owl:ObjectProperty})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$.

First, by the "$\rightarrow$" direction of RDFS semantic condition of ICEXT, we rewrite (1b) and (1c) to

$$(1b')\ I(\texttt{ex:x}) \in \text{ICEXT}(I(\texttt{owl:Class}))\ ,$$
$$(1c')\ I(\texttt{ex:x}) \in \text{ICEXT}(I(\texttt{owl:ObjectProperty}))\ .$$

By the "$\rightarrow$" direction of the semantic condition for class disjointness (OWL2/Tab5.9) and (1a) follows

$$(2)\ \forall z : \neg\,[z \in \text{ICEXT}(I(\texttt{owl:Class}))\ \wedge\ z \in \text{ICEXT}(I(\texttt{owl:ObjectProperty}))]\ .$$

Specialization (2) to $z := I(\texttt{ex:x})$ implies

$$(3)\ \neg\,[I(\texttt{ex:x}) \in \text{ICEXT}(I(\texttt{owl:Class}))\ \wedge\ I(\texttt{ex:x}) \in \text{ICEXT}(\texttt{owl:ObjectProperty}))]\ .$$

Now (3) is in contradiction with (1b') and (1c').

012_Template_Class (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Given the existence of individuals l_1, l_2, l_3 and r, such that

$(1a1)$ $\langle I(\texttt{foaf:Person}), I(\texttt{owl:Class})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$,
$(1b1)$ $\langle I(\texttt{ex:PersonAttribute}), l_1\rangle \in \text{IEXT}(I(\texttt{owl:intersectionOf}))$,
$(1c1)$ $\langle l_1, I(\texttt{owl:DatatypeProperty})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1c2)$ $\langle l_1, l_2\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1c3)$ $\langle l_2, I(\texttt{owl:FunctionalProperty})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1c4)$ $\langle l_2, l_3\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1c5)$ $\langle l_3, r\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1c6)$ $\langle l_3, I(\texttt{rdf:nil})\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1d1)$ $\langle r, I(\texttt{owl:Restriction})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$,
$(1d2)$ $\langle r, I(\texttt{rdfs:domain})\rangle \in \text{IEXT}(I(\texttt{owl:onProperty}))$,
$(1d3)$ $\langle r, I(\texttt{foaf:Person})\rangle \in \text{IEXT}(I(\texttt{owl:hasValue}))$,
$(1e1)$ $\langle I(\texttt{ex:name}), I(\texttt{ex:PersonAttribute})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$,
$(1f1)$ $\langle I(\texttt{ex:alice}), I(\texttt{"alice"})\rangle \in \text{IEXT}(I(\texttt{ex:name}))$.

From (1b1), (1c1) – (1c6), and the semantic condition for class intersection (OWL2/Tab5.4, "$\rightarrow$", ternary) follows

$$\begin{aligned}(2)\ \forall x : x \in{}& \text{ICEXT}(I(\texttt{ex:PersonAttribute})) \Leftrightarrow \\ & x \in \text{ICEXT}(I(\texttt{owl:DatatypeProperty})) \wedge \\ & x \in \text{ICEXT}(I(\texttt{owl:FunctionalProperty})) \wedge \\ & x \in \text{ICEXT}(r)\ .\end{aligned}$$

From (1e1) and the RDFS semantic condition for ICEXT ("→") follows

$$(3)\ I(\texttt{ex:name}) \in \mathrm{ICEXT}(I(\texttt{ex:PersonAttribute}))\ .$$

From (3) and (2) follows

$$(4)\ I(\texttt{ex:name}) \in \mathrm{ICEXT}(I(\texttt{owl:FunctionalProperty}))\ .$$

From (4) and the RDFS semantic condition for ICEXT ("←") follows

$$(5)\ \langle I(\texttt{ex:name}), I(\texttt{owl:FunctionalProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

From (1d2), (1d3) and the semantic condition for has-value restrictions (OWL2/Tab5.6) follows

$$\begin{aligned}(6)\ \forall x : x \in ICEXT(r) \Leftrightarrow \\ \langle x, I(\texttt{foaf:Person})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:domain}))\ .\end{aligned}$$

From (3) and (2) follows

$$(7)\ I(\texttt{ex:name}) \in \mathrm{ICEXT}(r)\ .$$

From (7) and (6) follows

$$(8)\ \langle I(\texttt{ex:name}), I(\texttt{foaf:Person})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:domain}))\ .$$

From (1f1), (8) and the RDFS semantic condition for property domains follows

$$(9)\ I(\texttt{ex:alice}) \in \mathrm{ICEXT}(I(\texttt{foaf:Person}))\ .$$

From (9) and the RDFS semantic condition for ICEXT ("←") follows

$$(10)\ \langle I(\texttt{ex:alice}), I(\texttt{foaf:Person})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

The conjecture follows from (5) and (10).

013_Cliques (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that

$I + B$ satisfies the premise graph. Let there be r, i, l_1, l_2 and l_3 such that

$$
\begin{array}{rl}
(1a) & \langle I(\texttt{ex:Clique}), I(\texttt{owl:Class})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ , \\
(1b) & \langle I(\texttt{ex:sameCliqueAs}), I(\texttt{owl:sameAs})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subPropertyOf}))\ , \\
(1c) & \langle I(\texttt{ex:sameCliqueAs}), I(\texttt{ex:Clique})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:range}))\ , \\
(1d) & \langle I(\texttt{ex:Clique}), r\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ , \\
(1e) & \langle r, I(\texttt{owl:Restriction})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ , \\
(1f) & \langle r, I(\texttt{ex:sameCliqueAs})\rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\ , \\
(1g) & \langle r, I(\texttt{ex:Clique})\rangle \in \mathrm{IEXT}(I(\texttt{owl:someValuesFrom}))\ , \\
(1h) & \langle I(\texttt{foaf:knows}), I(\texttt{owl:ObjectProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ , \\
(1j) & \langle I(\texttt{foaf:knows}), l_1\rangle \in \mathrm{IEXT}(I(\texttt{owl:propertyChainAxiom}))\ , \\
(1k) & \langle l_1, I(\texttt{rdf:type})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ , \\
(1m) & \langle l_1, l_2\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ , \\
(1n) & \langle l_2, I(\texttt{ex:sameCliqueAs})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ , \\
(1o) & \langle l_2, l_3\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ , \\
(1p) & \langle l_3, i\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ , \\
(1q) & \langle l_3, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ , \\
(1r) & \langle i, I(\texttt{rdf:type})\rangle \in \mathrm{IEXT}(I(\texttt{owl:inverseOf}))\ , \\
(1s) & \langle I(\texttt{ex:JoesGang}), I(\texttt{ex:Clique})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ , \\
(1t) & \langle I(\texttt{ex:alice}), I(\texttt{ex:JoesGang})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ , \\
(1u) & \langle I(\texttt{ex:bob}), I(\texttt{ex:JoesGang})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .
\end{array}
$$

From (1s) and the RDFS semantic condition of "ICEXT" ("→") follows

$$(1s')\ I(\texttt{ex:JoesGang}) \in \mathrm{ICEXT}(I(\texttt{ex:Clique}))\ .$$

From (1d) and the OWL 2 semantic condition of class subsumption (OWL2/Tab5.8, "→") follows

$$(2)\ \forall x:\ x \in \mathrm{ICEXT}(I(\texttt{ex:Clique})) \Rightarrow x \in \mathrm{ICEXT}(r)\ .$$

From (1f) – (1g) and the semantic condition for existential property restrictions (OWL2/Tab5.6) follows

$$
\begin{array}{l}
(3)\ \forall x : x \in \mathrm{ICEXT}(r) \Leftrightarrow \\
\qquad \exists y:\ \langle x, y\rangle \in \mathrm{IEXT}(I(\texttt{ex:sameCliqueAs}))\ \wedge\ y \in \mathrm{ICEXT}(I(\texttt{ex:Clique}))\ .
\end{array}
$$

From (1s'), (2) and (3) follows

$$(4)\ \exists y : \langle I(\texttt{ex:JoesGang}), y\rangle \in \mathrm{IEXT}(I(\texttt{ex:sameCliqueAs}))\ .$$

According to (4), we can find some y such that

$$(5)\ \langle I(\texttt{ex:JoesGang}), y\rangle \in \mathrm{IEXT}(I(\texttt{ex:sameCliqueAs}))\ .$$

By applying the OWL 2 semantic condition for property subsumption OWL2/Tab5.8, "↔") and (1b) to (5), we receive

$$(6)\ \langle I(\texttt{ex:JoesGang}), y\rangle \in \mathrm{IEXT}(I(\texttt{ex:sameAs}))\ .$$

Now, the semantic condition for owl:sameAs (OWL2/Tab5.9, "→") applied to (6) which yields

$$(6)\ y = I(\texttt{ex:JoesGang})\ .$$

By (4) and (6) we receive

$$(7)\ \langle I(\texttt{ex:JoesGang}), I(\texttt{ex:JoesGang})\rangle \in \mathrm{IEXT}(I(\texttt{ex:sameCliqueAs}))\ .$$

From (1r) and the semantic condition for inverse properties (OWL2/Tab5.12, "→") follows

$$(8)\ \forall x\ y :\ \langle x, y\rangle \in \mathrm{IEXT}(i) \Leftrightarrow \langle y, x\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

From (1u) and (8) follows:

$$(9)\ \langle I(\texttt{ex:JoesGang}), I(\texttt{ex:bob})\rangle \in \mathrm{IEXT}(i)\ .$$

From (1j) – (1q) and the semantic condition for sub property chains (OWL2/Tab5.11, "→", ternary) we receive

$$\begin{aligned}(10)\ \forall y_0,\ y_1,\ y_2,\ y_3 :&\\ &\langle y_0, y_1\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type})) \wedge\\ &\langle y_1, y_2\rangle \in \mathrm{IEXT}(I(\texttt{ex:sameCliqueAs})) \wedge\\ &\langle y_2, y_3\rangle \in \mathrm{IEXT}(i)\\ &\Rightarrow \langle y_0, y_3\rangle \in \mathrm{IEXT}(I(\texttt{foaf:knows}))\ .\end{aligned}$$

Finally, from (10), (1t), (7) and (9) follows

$$(11)\ \langle I(\texttt{ex:alice}), I(\texttt{ex:bob})\rangle \in \mathrm{IEXT}(I(\texttt{foaf:knows}))\ .$$

014_Harry_belongs_to_some_Species (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let u, l_1 and l_2 be individuals such that the following holds:

$$\begin{aligned}(1a)\ &\langle I(\texttt{ex:Eagle}), I(\texttt{ex:Species})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\ (1b)\ &\langle I(\texttt{ex:Falcon}), I(\texttt{ex:Species})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\ (1c)\ &\langle I(\texttt{ex:harry}), u\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\ (1d1)\ &\langle u, l_1\rangle \in \mathrm{IEXT}(I(\texttt{owl:unionOf}))\ ,\\ (1d2)\ &\langle l_1, I(\texttt{ex:Eagle})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\ (1d3)\ &\langle l_1, l_2\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\ (1d4)\ &\langle l_2, I(\texttt{ex:Falcon})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\ (1d5)\ &\langle l_2, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ .\end{aligned}$$

We prove the claim by classical dilemma.
Case 1: $\langle I(\texttt{ex:harry}), I(\texttt{ex:Eagle})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))$.
Then, together with (1a) follows:

$$\begin{aligned}(2)\ \exists x :\ &\langle I(\texttt{ex:harry}), x\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type})) \wedge\\ &\langle x, I(\texttt{ex:Species})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .\end{aligned}$$

Case 2: $\langle I(\texttt{ex:harry}), I(\texttt{ex:Eagle})\rangle \notin \mathrm{IEXT}(I(\texttt{rdf:type}))$.

By the RDFS semantic condition of ICEXT ("→") we get

$$(3)\ I(\texttt{ex:harry}) \notin \mathrm{ICEXT}(I(\texttt{ex:Eagle}))\ .$$

Likewise, from (1c) and the RDFS semantic condition of ICEXT ("→") we get

$$(4)\ I(\texttt{ex:harry}) \in \mathrm{ICEXT}(u)\ .$$

From (1d1) – (1d5) and the semantic condition of class union (OWL2/Tab5.4, "→", binary) follows

$$\begin{aligned}(5)\ &\forall x : x \in \mathrm{ICEXT}(u) \Leftrightarrow \\ &\quad x \in \mathrm{ICEXT}(I(\texttt{ex:Eagle})) \vee x \in \mathrm{ICEXT}(I(\texttt{ex:Falcon}))\ .\end{aligned}$$

Specializing (4) to $x := I(\texttt{ex:harry})$ implies

$$\begin{aligned}(6)\ &I(\texttt{ex:harry}) \in \mathrm{ICEXT}(u) \Leftrightarrow \\ &I(\texttt{ex:harry}) \in \mathrm{ICEXT}(\texttt{ex:Eagle}) \vee I(\texttt{ex:harry}) \in \mathrm{ICEXT}(\texttt{ex:Falcon})\ .\end{aligned}$$

By (4), (6) and (3) we get

$$(7)\ I(\texttt{ex:harry}) \in \mathrm{ICEXT}(I(\texttt{ex:Falcon}))\ .$$

Using the RDFS semantic condition of ICEXT ("←") results in

$$(8)\ \langle I(\texttt{ex:harry}), I(\texttt{ex:Falcon})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

From (8) and (1b) follows:

$$\begin{aligned}(9)\ \exists x : &\langle I(\texttt{ex:harry}), x\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type})) \wedge \\ &\langle x, I(\texttt{ex:Species})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .\end{aligned}$$

Since (2) and (9) are the same result from the contrary assumed cases, we get the claimed result.

015_Reflective_Tautologies_I (Proof) Let I be a satisfying OWL 2 RDF-Based interpretation for the empty graph. It is true that

$$I(\texttt{owl:sameAs}) = I(\texttt{owl:sameAs})\ .$$

By the "←" direction of semantic condition for `owl:sameAs` (OWL2/Tab5.9) we receive

$$\langle I(\texttt{owl:sameAs}), I(\texttt{owl:sameAs})\rangle \in \mathrm{IEXT}(I(\texttt{owl:sameAs}))\ .$$

016_Reflective_Tautologies_II (Proof) Let I be a satisfying OWL 2 RDF-Based interpretation for the empty graph. Given arbitrary c_1, c_2, and assume the following to hold:

$$(1)\ \langle c_1, c_2 \rangle \in \mathrm{IEXT}(I(\texttt{owl:equivalentClass}))\ .$$

From the "$\rightarrow$" direction of the semantic condition for class equivalence (OWL2/Tab5.9) and from the property extension of `owl:equivalentClass` (OWL2/Tab5.3) follows

$$\begin{aligned}&(2a)\ c_1 \in \mathrm{IC}\ ,\\&(2b)\ c_2 \in \mathrm{IC}\ ,\\&(2c)\ \forall x:\ x \in \mathrm{ICEXT}(c_1) \Leftrightarrow x \in \mathrm{ICEXT}(c_2)\ .\end{aligned}$$

From (2c) follows the weaker result

$$(3)\ \forall x:\ x \in \mathrm{ICEXT}(c_1) \Rightarrow x \in \mathrm{ICEXT}(c_2)\ .$$

From (2a), (2b) and (3) and the "$\leftarrow$" direction of the OWL 2 semantic condition of class subsumption (OWL2/Tab5.8) follows

$$(4)\ \langle c_1, c_2 \rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ .$$

Since (4) follows from (1) and since c_1 and c_2 were chosen arbitrarily, we get

$$\begin{aligned}(5)\ \forall x,\ y:\ &\langle x, y \rangle \in \mathrm{IEXT}(I(\texttt{owl:equivalentClass}))\\&\Rightarrow \langle x, y \rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ .\end{aligned}$$

For property `owl:equivalentClass` we receive from OWL2/Tab5.3

$$(6a)\ I(\texttt{owl:equivalentClass}) \in \mathrm{IP}\ .$$

and for property `rdfs:subClassOf` we receive from the RDFS axiomatic triples

$$(6b)\ \langle I(\texttt{rdfs:subClassOf}), I(\texttt{rdf:Property}) \rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

By the "$\leftarrow$" direction of the RDF semantic condition of IP and IEXT we receive from (6b):

$$(6b')\ I(\texttt{rdfs:subClassOf}) \in \mathrm{IP}\ .$$

From (5), (6a) and (6b') and from the "$\leftarrow$" direction of the OWL 2 semantic condition for property subsumption (OWL2/Tab5.8) we finally receive

$$(7)\ \langle I(\texttt{owl:equivalentClass}), I(\texttt{rdfs:subClassOf}) \rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subPropertyOf}))\ .$$

017_Builtin_Based_Definitions (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph. Let the following assertions hold:

$$\begin{aligned}&(1a)\ \langle I(\texttt{ex:notInstanceOf}), I(\texttt{rdf:type}) \rangle \in \mathrm{IEXT}(I(\texttt{owl:propertyDisjointWith}))\ ,\\&(1b)\ \langle I(\texttt{ex:w}), I(\texttt{ex:c}) \rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\&(1c)\ \langle I(\texttt{ex:u}), I(\texttt{ex:c}) \rangle \in \mathrm{IEXT}(I(\texttt{ex:notInstanceOf}))\ .\end{aligned}$$

From (1a) and the semantic condition for disjoint properties (OWL2/5.9, "$\rightarrow$") follows:

$$(1a')\ \forall x,\, y : \neg[\ \langle x,y\rangle \in \mathrm{IEXT}(I(\texttt{ex:notInstanceOf})) \wedge \langle x,y\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\,]\ .$$

Specializing (1a') to $x := I(\texttt{ex:w})$ and $y := I(\texttt{ex:c})$ implies

$$(2)\ \neg[\ \langle I(\texttt{ex:w}), I(\texttt{ex:c})\rangle \in \mathrm{IEXT}(I(\texttt{ex:notInstanceOf})) \wedge \langle I(\texttt{ex:w}), I(\texttt{ex:c})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\,]\ .$$

From (1b) and (2) follows

$$(3)\ \langle I(\texttt{ex:w}), I(\texttt{ex:c})\rangle \notin \mathrm{IEXT}(I(\texttt{ex:notInstanceOf}))\ .$$

From (1c) and (3) follows

$$(4)\ I(\texttt{ex:w}) \neq I(\texttt{ex:u})\ .$$

From (4) and the semantic condition for `owl:differentFrom` (OWL2/Tab5.9, "$\leftarrow$") follows:

$$(5)\langle I(\texttt{ex:w}), I(\texttt{ex:u})\rangle \in \mathrm{IEXT}(I(\texttt{owl:differentFrom}))\ .$$

018_Modified_Logical_Vocabulary_Semantics (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph. We start from:

$$(1a)\ \langle I(\texttt{owl:sameAs}), I(\texttt{ex:Person})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:domain}))\ ,$$
$$(1b)\ \langle I(\texttt{ex:w}), I(\texttt{ex:u})\rangle \in \mathrm{IEXT}(I(\texttt{owl:sameAs}))\ .$$

From this we get via the RDFS semantic condition for property domains:

$$(2)\ I(\texttt{ex:w}) \in \mathrm{ICEXT}(I(\texttt{ex:Person}))\ .$$

Further, from (1b) and the semantic condition for `owl:sameAs` (OWL2/Tab5.9, "$\rightarrow$") we get

$$(3)\ I(\texttt{ex:w}) = I(\texttt{ex:u})\ .$$

This allows for substitution in (2), providing

$$(4)\ I(\texttt{ex:u}) \in \mathrm{ICEXT}(I(\texttt{ex:Person}))\ .$$

Finally, by applying the RDFS semantic extension for ICEXT ("$\leftarrow$") to (4) we get

$$(5)\ \langle I(\texttt{ex:u}), I(\texttt{ex:Person})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

019_Disjoint_Annotation_Properties (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the premise graph. Starting from:

$$(1a)\ \langle I(\texttt{skos:prefLabel}), I(\texttt{skos:altLabel})\rangle \in \mathrm{IEXT}(I(\texttt{owl:propertyDisjointWith}))\ ,$$
$$(1b)\ \langle I(\texttt{ex:foo}), I(\texttt{"foo"})\rangle \in \mathrm{IEXT}(I(\texttt{skos:prefLabel}))\ ,$$
$$(1c)\ \langle I(\texttt{ex:foo}), I(\texttt{"foo"})\rangle \in \mathrm{IEXT}(I(\texttt{skos:altLabel}))\ .$$

From (1a) and the semantic condition of property disjointness (OWL2/Tab5.9, "$\rightarrow$") we receive

$$(2)\ \forall x, y :\ \neg[\ \langle x, y\rangle \in \mathrm{IEXT}(I(\texttt{skos:prefLabel})) \wedge \langle x, y\rangle \in \mathrm{IEXT}(I(\texttt{skos:altLabel}))\ ]\ .$$

Specifically, we receive:

$$(3)\ \neg[\ \langle I(\texttt{ex:foo}), I(\texttt{"foo"})\rangle \in \mathrm{IEXT}(I(\texttt{skos:prefLabel})) \wedge \langle I(\texttt{ex:foo}), I(\texttt{"foo"})\rangle \in \mathrm{IEXT}(I(\texttt{skos:altLabel}))\ ]\ .$$

We now have a contradiction between (1b), (1c) and (3).

020_Logical_Complications (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be xs, xc, lu_1, lu_2, lu_3, li_1, li_2, li_3, such that

$$(1a)\ \langle I(\texttt{ex:c}), lu_1\rangle \in \mathrm{IEXT}(I(\texttt{owl:unionOf}))\ ,$$
$$(1b)\ \langle lu_1, I(\texttt{ex:c1})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,$$
$$(1c)\ \langle lu_1, lu_2\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,$$
$$(1d)\ \langle lu_2, I(\texttt{ex:c2})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,$$
$$(1e)\ \langle lu_2, lu_3\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,$$
$$(1f)\ \langle lu_3, I(\texttt{ex:c3})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,$$
$$(1g)\ \langle lu_3, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,$$
$$(1h)\ \langle I(\texttt{ex:d}), I(\texttt{ex:c1})\rangle \in \mathrm{IEXT}(I(\texttt{owl:disjointWith}))\ ,$$
$$(1j)\ \langle I(\texttt{ex:d}), xs\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ ,$$
$$(1k)\ \langle xs, li_1\rangle \in \mathrm{IEXT}(I(\texttt{owl:intersectionOf}))\ ,$$
$$(1m)\ \langle li_1, I(\texttt{ex:c})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,$$
$$(1n)\ \langle li_1, li_2\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,$$
$$(1o)\ \langle li_2, xc\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,$$
$$(1p)\ \langle li_2, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,$$
$$(1q)\ \langle xc, I(\texttt{ex:c2})\rangle \in \mathrm{IEXT}(I(\texttt{owl:complementOf}))\ .$$

From (1a), (1b) – (1g) and the semantic condition for class union (OWL2/Tab.5.4, "$\rightarrow$", ternary) follows

$$(2)\ I(\texttt{ex:c3}) \in \mathrm{IC}\ .$$

and

$$(3)\ \forall x : x \in \mathrm{ICEXT}(I(\texttt{ex:c})) \Leftrightarrow x \in \mathrm{ICEXT}(I(\texttt{ex:c1}))\ \vee\ x \in \mathrm{ICEXT}(I(\texttt{ex:c2}))\ \vee\ x \in \mathrm{ICEXT}(I(\texttt{ex:c3}))\ .$$

From (1h) and the semantic condition for class disjointness follows

$$(4)\ I(\texttt{ex:d}) \in \mathrm{IC}$$

and from (1h) and the semantic condition for class disjointness (OWL2/Tab.5.9, "$\rightarrow$") follows

$$(5)\ \forall x :\ \neg[\, x \in \mathrm{ICEXT}(I(\texttt{ex:d}))\ \wedge\ x \in \mathrm{ICEXT}(I(\texttt{ex:c1}))\,]\ .$$

From (1j) and the OWL 2 semantic condition of class subsumption (OWL2/Tab5.9, "$\rightarrow$") follows

$$(6)\ \forall x :\ x \in \mathrm{ICEXT}(I(\texttt{ex:d})) \Rightarrow x \in \mathrm{ICEXT}(xs)\ .$$

From (1k), (1m) – (1p) and the semantic condition for class intersection (OWL2/Tab5.4, "$\rightarrow$", binary) follows

$$(7)\ \forall x :\ x \in \mathrm{ICEXT}(xs) \Leftrightarrow x \in \mathrm{ICEXT}(I(\texttt{ex:c}))\ \wedge\ x \in \mathrm{ICEXT}(xc)\ .$$

From (1q) and the semantic condition for class complement (OWL2/Tab5.4, "$\rightarrow$") follows

$$(8)\ \forall x :\ x \in ICEXT(xc) \Leftrightarrow x \notin \mathrm{ICEXT}(I(\texttt{ex:c2}))\ .$$

From (6), (7) and (8) follows

$$(9)\ \forall x :\ x \in \mathrm{ICEXT}(I(\texttt{ex:d})) \Rightarrow x \in \mathrm{ICEXT}(I(\texttt{ex:c}))\ \wedge\ x \notin \mathrm{ICEXT}(I(\texttt{ex:c2}))\ .$$

From (9) and (3) follows

$$(10)\ \forall x :\ x \in \mathrm{ICEXT}(I(\texttt{ex:d})) \Rightarrow x \in \mathrm{ICEXT}(I(\texttt{ex:c1}))\ \vee\ x \in \mathrm{ICEXT}(I(\texttt{ex:c3}))\ .$$

From (10) and (5) follows

$$(11)\ \forall x :\ x \in \mathrm{ICEXT}(I(\texttt{ex:d})) \Rightarrow x \in \mathrm{ICEXT}(I(\texttt{ex:c3}))\ .$$

Finally, from (4), (2), (11) and the OWL 2 semantic extension of class subsumption (OWL2/Tab5.8, "$\leftarrow$") follows

$$(12)\ \langle I(\texttt{ex:d}), I(\texttt{ex:c3})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ .$$

021_Composite_Enumerations (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be l_{11}, l_{12}, l_{21}, l_{22},

l_{31}, l_{32}, l_{41}, l_{42}, such that the following statements hold:

$$
\begin{aligned}
&(1a1)\ \langle I(\texttt{ex:c1}), l_{11}\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
&(1a2)\ \langle l_{11}, I(\texttt{ex:w1})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1a3)\ \langle l_{11}, l_{12})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1a4)\ \langle l_{12}, I(\texttt{ex:w2})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1a5)\ \langle l_{12}, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1b1)\ \langle I(\texttt{ex:c2}), l_{21})\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
&(1b2)\ \langle l_{21}, I(\texttt{ex:w2})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1b3)\ \langle l_{21}, l_{22})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1b4)\ \langle l_{22}, I(\texttt{ex:w3})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1b5)\ \langle l_{22}, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1c1)\ \langle I(\texttt{ex:c3}), l_{31})\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
&(1c2)\ \langle l_{31}, I(\texttt{ex:w1})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1c3)\ \langle l_{31}, l_{32}\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1c4)\ \langle l_{32}, I(\texttt{ex:w2})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1c5)\ \langle l_{32}, l_{33}\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1c6)\ \langle l_{33}, I(\texttt{ex:w3})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1c7)\ \langle l_{33}, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1d1)\ \langle I(\texttt{ex:c4}), l_{41}\rangle \in \mathrm{IEXT}(I(\texttt{owl:unionOf}))\ ,\\
&(1d2)\ \langle l_{41}, I(\texttt{ex:c1})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1d3)\ \langle l_{41}, l_{42})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
&(1d4)\ \langle l_{42}, I(\texttt{ex:c2})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1d5)\ \langle l_{42}, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ .
\end{aligned}
$$

By applying the semantic condition for enumeration classes (OWL2/Tab5.5, "→", binary and ternary) to (1a1)–(1a5), (1b1)–(1b5) and (1c1)–(1c7), respectively, we receive:

$$
\begin{aligned}
&(2a)\ \forall x:\ x \in \mathrm{ICEXT}(I(\texttt{ex:c1})) \Leftrightarrow x = I(\texttt{ex:w1}) \vee x = I(\texttt{ex:w2})\ ,\\
&(2b)\ \forall x:\ x \in \mathrm{ICEXT}(I(\texttt{ex:c2})) \Leftrightarrow x = I(\texttt{ex:w2}) \vee x = I(\texttt{ex:w3})\ ,\\
&(2c)\ \forall x:\ x \in \mathrm{ICEXT}(I(\texttt{ex:c3})) \Leftrightarrow x = I(\texttt{ex:w1}) \vee x = I(\texttt{ex:w2}) \vee x = I(\texttt{ex:w3})\ .
\end{aligned}
$$

By applying the semantic condition for class union (OWL2/Tab5.4, "→", binary) to (1d1)–(1d5), we receive:

$$(2d)\ \forall x:\ x \in \mathrm{ICEXT}(\texttt{ex:c4}) \Leftrightarrow x \in \mathrm{ICEXT}(I(\texttt{ex:c1})) \vee x \in \mathrm{ICEXT}(I(\texttt{ex:c2}))\ .$$

From the property extension of `owl:oneOf` (OWL2/Tab5.3) and (1c1) follows

$$(3a)\ I(\texttt{ex:c3}) \in \mathrm{IC}\ .$$

From the property extension of `owl:unionOf` (OWL2/Tab5.3) and (1d1) follows

$$(3b)\ I(\texttt{ex:c4}) \in \mathrm{IC}\ .$$

From (2a), (2b) and (2c) follows

$$(4)\ \forall x:\ x \in \mathrm{ICEXT}(I(\texttt{ex:c3})) \Leftrightarrow x \in \mathrm{ICEXT}(I(\texttt{ex:c1})) \vee x \in \mathrm{ICEXT}(I(\texttt{ex:c2}))\ .$$

From (2d) and (4) follows

$$(5)\ \forall x:\ x \in \text{ICEXT}(I(\texttt{ex:c3})) \Leftrightarrow x \in \text{ICEXT}(I(\texttt{ex:c4}))\ .$$

From the semantic condition for class equivalence (OWL2/Tab5.9, ←), (3a), (3b) and (5) follows

$$(6)\ \langle I(\texttt{ex:c3}), I(\texttt{ex:c4})\rangle \in \text{IEXT}(I(\texttt{owl:equivalentClass})).$$

022_List_Member_Access (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Given an individual pL as well as list individuals l_{11}, l_{12}, l_{21}, l_{22}, l_{31}, l_{32} and l_{33}, such that the following assertions hold:

$(1a1)\ \langle I(\texttt{skos:memberList}), pL\rangle \in \text{IEXT}(I(\texttt{rdfs:subPropertyOf}))$,
$(1b1)\ \langle I(\texttt{skos:member}), l_{11}\rangle \in \text{IEXT}(I(\texttt{owl:propertyChainAxiom}))$,
$(1b2)\ \langle l_{11}, pL\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1b3)\ \langle l_{11}, l_{12}\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1b4)\ \langle l_{12}, I(\texttt{rdf:first})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1b5)\ \langle l_{12}, I(\texttt{rdf:nil})\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1c1)\ \langle pL, l_{21}\rangle \in \text{IEXT}(I(\texttt{owl:propertyChainAxiom}))$,
$(1c2)\ \langle l_{21}, pL\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1c3)\ \langle l_{21}, l_{22}\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1c4)\ \langle l_{22}, I(\texttt{rdf:rest})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1c5)\ \langle l_{22}, I(\texttt{rdf:nil})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1d1)\ \langle I(\texttt{ex:MyOrderedCollection}), I(\texttt{skos:OrderedCollection})\rangle \in \text{IEXT}(I(\texttt{rdf:type}))$,
$(1e1)\ \langle I(\texttt{ex:MyOrderedCollection}), l_{31}\rangle \in \text{IEXT}(I(\texttt{skos:memberList}))$,
$(1e2)\ \langle l_{31}, I(\texttt{ex:X})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1e3)\ \langle l_{31}, l_{32})\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1e4)\ \langle l_{32}, I(\texttt{ex:Y})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1e5)\ \langle l_{32}, l_{33})\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$,
$(1e6)\ \langle l_{33}, I(\texttt{ex:Z})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))$,
$(1e7)\ \langle l_{33}, I(\texttt{rdf:nil})\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))$.

By the RDFS semantic condition for property subsumption, (1a1) and (1e1) we receive

$$(1e1')\ \langle I(\texttt{ex:MyOrderedCollection}), l_{31}\rangle \in \text{IEXT}(pL)\ .$$

From the semantic condition of sub property chains (OWL2/Tab5.11, "→", binary version) we get from (1b1) to (1b5):

$$\begin{aligned}(2b)\ \forall y_0, y_1, y_2 :\ & \langle y_0, y_1\rangle \in \text{IEXT}(pL) \wedge \\ & \langle y_1, y_2\rangle \in \text{IEXT}(I(\texttt{rdf:first})) \\ & \Rightarrow \langle y_0, y_2\rangle \in \text{IEXT}(I(\texttt{skos:member}))\ .\end{aligned}$$

and from (1c1) to (1c5) we get

$$\begin{aligned}(2c)\ \forall y_0, y_1, y_2 :\ & \langle y_0, y_1\rangle \in \text{IEXT}(pL) \wedge \\ & \langle y_1, y_2\rangle \in \text{IEXT}(I(\texttt{rdf:rest})) \\ & \Rightarrow \langle y_0, y_2\rangle \in \text{IEXT}(pL)\ .\end{aligned}$$

We receive the first result triple

$$(3a)\ \langle I(\texttt{ex:MyOrderedCollection}), I(\texttt{ex:X})\rangle \in \mathrm{IEXT}(I(\texttt{skos:member}))\ .$$

from (2b), (1e1') and (1e2). Further, by (2c), (1e1') and (1e3) we receive

$$(1e1'')\ \langle I(\texttt{ex:MyOrderedCollection}), l_{32}\rangle \in \mathrm{IEXT}(pL)\ .$$

In the same way, from (2b), (1e1"), (1e4) and (1e5) we receive

$$(3b)\ \langle I(\texttt{ex:MyOrderedCollection}), I(\texttt{ex:Y})\rangle \in \mathrm{IEXT}(I(\texttt{skos:member}))\ .$$

and

$$(1e1''')\ \langle I(\texttt{ex:MyOrderedCollection}), l_{33}\rangle \in \mathrm{IEXT}(pL)\ .$$

And likewise, we receive

$$(3c)\ \langle I(\texttt{ex:MyOrderedCollection}), I(\texttt{ex:Z})\rangle \in \mathrm{IEXT}(I(\texttt{skos:member}))\ .$$

023_Unique_List_Components (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be individuals o and l such that

$$\begin{aligned}
&(1a)\ \langle I(\texttt{rdf:first}), I(\texttt{owl:FunctionalProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1b)\ \langle I(\texttt{ex:w}), o\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1c)\ \langle o, I(\texttt{owl:Class})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
&(1d)\ \langle o, l\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
&(1e)\ \langle l, I(\texttt{ex:u})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1f)\ \langle l, I(\texttt{ex:v})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
&(1g)\ \langle l, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ .
\end{aligned}$$

By the RDFS semantic condition for ICEXT and (1b) follows

$$(1b')\ I(\texttt{ex:w}) \in \mathrm{ICEXT}(o)\ .$$

By the semantic condition for enumeration classes (OWL2/Tab5.5, "→", singleton) and (1d), (1e) and (1g) follows

$$(2)\ \forall x:\ x \in \mathrm{ICEXT}(o) \Leftrightarrow x = I(\texttt{ex:u})\ .$$

From (1b') and specializing (2) to $x := I(\texttt{ex:w})$ follows

$$(3)\ I(\texttt{ex:w}) = I(\texttt{ex:u})\ .$$

By the semantic condition for functional properties (OWL2/Tab5.13, "→") and (1a) follows

$$\begin{aligned}(4)\ \forall x, y1, y2 :\ &\langle x, y1\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first})) \wedge\\ &\langle x, y2\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\\ &\Rightarrow y1 = y2\ .\end{aligned}$$

From (1e), (1f) and from specializing (4) to $x := l$ follows

$$(5)\ I(\texttt{ex:u}) = I(\texttt{ex:v})\ .$$

From the semantic condition of `owl:sameAs` (OWL2/Tab5.9, "←") and (3) and (5) follows

$$(6a)\ \langle I(\texttt{ex:w}), I(\texttt{ex:u})\rangle \in \mathrm{IEXT}(I(\texttt{owl:sameAs}))\ ,$$
$$(6b)\ \langle I(\texttt{ex:w}), I(\texttt{ex:v})\rangle \in \mathrm{IEXT}(I(\texttt{owl:sameAs}))\ .$$

024_Cardinality_Restrictions_on_Complex_Properties (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be z such that

$$(1a)\ \langle I(\texttt{ex:hasAncestor}), I(\texttt{owl:TransitiveProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,$$
$$(1b)\ \langle I(\texttt{ex:Person}), z\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ ,$$
$$(1c)\ \langle z, I(\texttt{owl:Restriction})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,$$
$$(1d)\ \langle z, I(\texttt{ex:hasAncestor})\rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\ ,$$
$$(1e)\ \langle z, I(\texttt{"1"\^{}\^{}xsd:nonNegativeInteger})\rangle \in \mathrm{IEXT}(I(\texttt{owl:minCardinality}))\ ,$$
$$(1f)\ \langle I(\texttt{ex:alice}), I(\texttt{ex:Person})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,$$
$$(1g)\ \langle I(\texttt{ex:bob}), I(\texttt{ex:Person})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,$$
$$(1h)\ \langle I(\texttt{ex:alice}), I(\texttt{ex:bob})\rangle \in \mathrm{IEXT}(I(\texttt{ex:hasAncestor}))\ .$$

From (1a) and the semantic condition of transitive properties (OWL2/Tab5.13, "→") follows

$$\begin{aligned}(2)\ \forall y_1, y_2, y_3 : {} & \langle y_1, y_2\rangle \in \mathrm{IEXT}(I(\texttt{ex:hasAncestor})) \wedge \\ & \langle y_2, y_3\rangle \in \mathrm{IEXT}(I(\texttt{ex:hasAncestor})) \\ & \Rightarrow \langle y_1, y_3\rangle \in \mathrm{IEXT}(I(\texttt{ex:hasAncestor}))\ .\end{aligned}$$

From (1b) and the OWL 2 semantic condition for class subsumption (OWL2/Tab5.8, "→") follows

$$(3)\ \forall y :\ y \in \mathrm{ICEXT}(I(\texttt{ex:Person})) \Rightarrow y \in \mathrm{ICEXT}(z)\ .$$

From (1c)–(1e) and the semantic condition for 1-min cardinality restrictions (OWL2/Tab5.6) follows

$$(4)\ \forall y :\ y \in \mathrm{ICEXT}(z) \Leftrightarrow \exists x :\ \langle y, x\rangle \in \mathrm{ICEXT}(I(\texttt{ex:hasAncestor}))\ .$$

From (3) and (4) follows

$$(5)\ \forall y :\ \mathrm{ICEXT}(\texttt{ex:Person}, y) \Rightarrow \exists x :\ \langle y, x\rangle \in \mathrm{IEXT}(\texttt{ex:hasAncestor})\ .$$

Applying the RDFS semantic condition for ICEXT ("→") to (1g) yields

$$(6)\ I(\texttt{ex:bob}) \in \mathrm{ICEXT}(I(\texttt{ex:Person}))\ .$$

From (6) and (5) follows the existence of some x such that

$$(7)\ \langle I(\texttt{ex:bob}), x\rangle \in \text{IEXT}(I(\texttt{ex:hasAncestor}))\ .$$

From (1h), (7) and (2) follows, for the same x,

$$(8)\ \langle I(\texttt{ex:alice}), x\rangle \in \text{IEXT}(I(\texttt{ex:hasAncestor}))\ .$$

Hence we have shown in (7) and (8) that

$$\begin{aligned}(9)\ \exists x : {} & \langle I(\texttt{ex:bob}), x\rangle \in \text{IEXT}(I(\texttt{ex:hasAncestor})) \wedge \\ & \langle I(\texttt{ex:alice}), x\rangle \in \text{IEXT}(I(\texttt{ex:hasAncestor}))\ .\end{aligned}$$

Therefore, there is some blank node mapping B' for the blank nodes in the conclusion graph such that $I + B'$ satisfies the conclusion graph.

025_Cyclic_Dependencies_between_Complex_Properties (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be individuals l_{11}, l_{12}, l_{21}, l_{22}, l_3, such that:

$$\begin{aligned}
(1a1)\ & \langle I(\texttt{ex:hasUncle}), l_{11}\rangle \in \text{IEXT}(I(\texttt{owl:propertyChainAxiom}))\ , \\
(1a2)\ & \langle l_{11}, I(\texttt{ex:hasCousin})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))\ , \\
(1a3)\ & \langle l_{11}, l_{12}\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))\ , \\
(1a4)\ & \langle l_{12}, I(\texttt{ex:hasFather})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))\ , \\
(1a5)\ & \langle l_{12}, I(\texttt{rdf:nil})\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))\ , \\
(1b1)\ & \langle I(\texttt{ex:hasCousin}), l_{21}\rangle \in \text{IEXT}(I(\texttt{owl:propertyChainAxiom}))\ , \\
(1b2)\ & \langle l_{21}, I(\texttt{ex:hasUncle})\rangle \in \text{IEXT}(I(\texttt{rdf:first}))\ , \\
(1b3)\ & \langle l_{21}, l_{22}\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))\ , \\
(1b4)\ & \langle l_{22}, l_3\rangle \in \text{IEXT}(I(\texttt{rdf:first}))\ , \\
(1b5)\ & \langle l_{22}, I(\texttt{rdf:nil})\rangle \in \text{IEXT}(I(\texttt{rdf:rest}))\ , \\
(1c)\ & \langle l_3, I(\texttt{ex:hasFather})\rangle \in \text{IEXT}(I(\texttt{owl:inverseOf}))\ , \\
(1d)\ & \langle I(\texttt{ex:alice}), I(\texttt{ex:dave})\rangle \in \text{IEXT}(I(\texttt{ex:hasFather}))\ , \\
(1e)\ & \langle I(\texttt{ex:alice}), I(\texttt{ex:bob})\rangle \in \text{IEXT}(I(\texttt{ex:hasCousin}))\ , \\
(1f)\ & \langle I(\texttt{ex:bob}), I(\texttt{ex:charly})\rangle \in \text{IEXT}(I(\texttt{ex:hasFather}))\ , \\
(1g)\ & \langle I(\texttt{ex:bob}), I(\texttt{ex:dave})\rangle \in \text{IEXT}(I(\texttt{ex:hasUncle}))\ .
\end{aligned}$$

From the semantic condition for sub property chains (OWL2/Tab5.11, "→", binary) and (1a1) to (1a5) follows

$$\begin{aligned}(2a)\ \forall y_0,\, y_1,\, y_2 : {} & \langle y_0, y_1\rangle \in \text{IEXT}(I(\texttt{ex:hasCousin})) \wedge \\ & \langle y_1, y_2\rangle \in \text{IEXT}(I(\texttt{ex:hasFather})) \\ & \Rightarrow \langle y_0, y_2\rangle \in \text{IEXT}(I(\texttt{ex:hasUncle}))\ .\end{aligned}$$

From the semantic condition for sub property chains (OWL2/Tab5.11, "→", binary) and (1b1) to (1b5) follows

$$\begin{aligned}(2b)\ \forall y_0,\, y_1,\, y_2 : {} & \langle y_0, y_1\rangle \in \text{IEXT}(I(\texttt{ex:hasUncle})) \wedge \\ & \langle y_1, y_2\rangle \in \text{IEXT}(l_3) \\ & \Rightarrow \langle y_0, y_2\rangle \in \text{IEXT}(I(\texttt{ex:hasCousin}))\ .\end{aligned}$$

From the semantic condition for inverse properties (OWL2/Tab5.12, "→") and (1c) follows

$$(2c)\ \forall x,\, y:\ \langle x,y\rangle \in \mathrm{IEXT}(l_3) \Leftrightarrow \langle y,x\rangle \in \mathrm{IEXT}(I(\texttt{ex:hasFather}))\ .$$

From (2c) and (1d) follows

$$(1d')\ \langle I(\texttt{ex:dave}), I(\texttt{ex:alice})\rangle \in \mathrm{IEXT}(l_3)\ .$$

From (2a), (1e) and (1f) follows

$$(3a)\ \langle I(\texttt{ex:alice}), I(\texttt{ex:charly})\rangle \in \mathrm{IEXT}(I(\texttt{ex:hasUncle}))\ .$$

From (2b), (1g) and (1d')

$$(3b)\ \langle I(\texttt{ex:bob}), I(\texttt{ex:alice})\rangle \in \mathrm{IEXT}(I(\texttt{ex:hasCousin}))\ .$$

The resulting triples are (3a) and (3b).

026_Inferred_Property_Characteristics_I (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let x_1, x_2, l_1 and l_2 be individuals such that the following holds:

$$\begin{array}{l}
(1a)\ \langle I(\texttt{ex:p}), x_1\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:domain}))\ ,\\
(1b)\ \langle x_1, l_1\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
(1c)\ \langle l_1, I(\texttt{ex:w})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
(1d)\ \langle l_1, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
(1e)\ \langle I(\texttt{ex:p}), x_2\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:range}))\ ,\\
(1f)\ \langle x_2, l_2\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
(1g)\ \langle l_2, I(\texttt{ex:u})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
(1h)\ \langle l_2, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ .
\end{array}$$

From the semantic condition for enumeration classes (OWL2/Tab5.5, "→", singleton) and (1b) to (1d) follows:

$$(2)\ \forall z:\ z \in \mathrm{ICEXT}(x_1) \Leftrightarrow z = I(\texttt{ex:w})\ .$$

Now let us assume that there are s_1, s_2 and o such that

$$\begin{array}{l}
(3a)\ \langle s_1, o\rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))\ ,\ \text{and}\\
(3b)\ \langle s_2, o\rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))\ .
\end{array}$$

From the RDFS semantic condition for property domains, (1a) and (3a) follows

$$(4a)\ s_1 \in \mathrm{ICEXT}(x_1)\ .$$

From the RDFS semantic condition for property domains, (1a) and (3b) follows

$$(4b)\ s_2 \in \mathrm{ICEXT}(x_1)\ .$$

From (2) and (4a) follows

$$(5a)\ s_1 = I(\texttt{ex:w})\ .$$

From (2) and (4b) follows

$$(5b)\ s_2 = I(\texttt{ex:w})\ .$$

Finally, from (5a) and (5b) follows

$$(6)\ s_1 = s_2\ .$$

Since s_1, s_2 and o were chosen arbitrarily, we can generalize

$$\begin{aligned}(7)\ \forall s_1,\ s_2,\ o : &\langle s_1, o\rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \wedge \\ &\langle s_2, o\rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \\ &\Rightarrow s_1 = s_2\ .\end{aligned}$$

From the RDFS axiomatic triple for the domain of `rdfs:domain`, we receive

$$(8)\ \langle I(\texttt{rdfs:domain}), I(\texttt{rdf:Property})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:domain}))\ .$$

From the RDFS semantic condition for property domains, (8) and (1a) follows

$$(9)\ I(\texttt{ex:p}) \in \mathrm{ICEXT}(I(\texttt{rdf:Property}))\ .$$

With the RDFS semantic condition for ICEXT and the RDF semantic condition for IP and IEXT follows

$$(9')\ I(\texttt{ex:p}) \in \mathrm{IP}\ .$$

By the semantic condition for inverse-functional properties (OWL2/Tab5.13, "←"), (9') and (7) follows

$$(10)\ I(\texttt{ex:p}) \in \mathrm{ICEXT}(I(\texttt{owl:InverseFunctionalProperty}))\ .$$

And by the RDFS semantic condition for ICEXT and (10) follows

$$(10')\ \langle I(\texttt{ex:p}), I(\texttt{owl:InverseFunctionalProperty})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

027_Inferred_Property_Characteristics_II (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be l_1, l_2 and v, such that

$$\begin{aligned}&(1a)\ \langle I(\texttt{owl:sameAs}), l_1\rangle \in \mathrm{IEXT}(I(\texttt{owl:propertyChainAxiom}))\ ,\\ &(1b)\ \langle l_1, I(\texttt{ex:p})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\ &(1c)\ \langle l_1, l_2\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\ &(1d)\ \langle l_2, v\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\ &(1e)\ \langle l_2, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\ &(1f)\ \langle v, I(\texttt{ex:p})\rangle \in \mathrm{IEXT}(I(\texttt{owl:inverseOf}))\ .\end{aligned}$$

From (1a), (1b) – (1e) and the semantic condition for sub property chains (OWL2/Tab5.11, "→", binary) follows

$$(2)\ I(\texttt{ex:p}) \in \mathrm{IP}$$

and

$$\begin{aligned}(3)\ \forall y_0,\, y_1,\, y_2 :\ & \langle y_0, y_1 \rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \wedge \\ & \langle y_1, y_2 \rangle \in \mathrm{IEXT}(v) \\ & \Rightarrow \langle y_0, y_2 \rangle \in \mathrm{IEXT}(I(\texttt{owl:sameAs}))\ .\end{aligned}$$

From (1f) and the semantic condition for inverse properties (OWL2/Tab5.12, "→") follows:

$$(4)\ \forall x,\, y :\ \langle x, y \rangle \in \mathrm{IEXT}(v) \Leftrightarrow \langle y, x \rangle \in \mathrm{IEXT}(I(\texttt{ex:p}))\ .$$

From the semantic condition for `owl:sameAs` (OWL2/Tab5.9, "→") follows:

$$(5)\ \forall x,\, y :\ \langle x, y \rangle \in \mathrm{IEXT}(I(\texttt{owl:sameAs})) \Rightarrow x = y\ .$$

From (3), (4) and (5) follows

$$\begin{aligned}(6)\ \forall y_0,\, y_1,\, y_2 :\ & \langle y_0, y_1 \rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \wedge \\ & \langle y_2, y_1 \rangle \in \mathrm{IEXT}(I(\texttt{ex:p})) \\ & \Rightarrow y_0 = y_2\ .\end{aligned}$$

From (2), (6) and the semantic condition for inverse-functional properties (OWL2/Tab.13, "←") follows

$$(7)\ I(\texttt{ex:p}) \in \mathrm{ICEXT}(I(\texttt{owl:InverseFunctionalProperty}))\ .$$

Finally, from (7) and the RDFS semantic condition for ICEXT ("←") follows

$$(8)\ \langle I(\texttt{ex:p}), I(\texttt{owl:InverseFunctionalProperty}) \rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

028_Inferred_Property_Characteristics_III (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be a z such that

$$\begin{aligned}&(1a)\ \langle I(\texttt{ex:InversesOfFunctionalProperties}), z \rangle \in \mathrm{IEXT}(I(\texttt{owl:equivalentClass}))\ ,\\ &(1b)\ \langle z, I(\texttt{owl:Restriction}) \rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\ &(1c)\ \langle z, I(\texttt{owl:inverseOf}) \rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\ ,\\ &(1d)\ \langle z, I(\texttt{owl:FunctionalProperty}) \rangle \in \mathrm{IEXT}(I(\texttt{owl:someValuesFrom}))\ .\end{aligned}$$

From (1a) and the semantic condition for class equivalence (OWL2/Tab5.9, "→") follows

$$(2)\ \forall x :\ x \in \mathrm{ICEXT}(I(\texttt{ex:InversesOfFunctionalProperties})) \Leftrightarrow x \in \mathrm{ICEXT}(z)\ .$$

From (1b) – (1d) and the semantic condition for existential property restrictions (OWL2/Tab5.6) follows

$$(3)\ \forall x : x \in \mathrm{ICEXT}(z) \Leftrightarrow \\ \exists y :\ [\,\langle x,y\rangle \in \mathrm{IEXT}(I(\texttt{owl:inverseOf})) \wedge \\ y \in \mathrm{ICEXT}(I(\texttt{owl:FunctionalProperty}))\,]\ .$$

From (2) and (3) follows

$$(4)\ \forall x : x \in \mathrm{ICEXT}(I(\texttt{ex:InversesOfFunctionalProperties})) \Leftrightarrow \\ \exists y :\ [\,\langle x,y\rangle \in \mathrm{IEXT}(I(\texttt{owl:inverseOf})) \wedge \\ y \in \mathrm{ICEXT}(I(\texttt{owl:FunctionalProperty}))\,]\ .$$

Let p be an arbitrary individual such that

$$(5)\ p \in \mathrm{ICEXT}(I(\texttt{ex:InversesOfFunctionalProperties}))\ .$$

We receive (5) and (4):

$$(6)\ \exists y : \langle p,y\rangle \in \mathrm{IEXT}(I(\texttt{owl:inverseOf})) \wedge \\ y \in \mathrm{ICEXT}(I(\texttt{owl:FunctionalProperty}))\ .$$

According to (6) there is a q for p such that

$$(7)\ \langle p,q\rangle \in \mathrm{IEXT}(I(\texttt{owl:inverseOf})) \wedge \\ q \in \mathrm{ICEXT}(I(\texttt{owl:FunctionalProperty}))\ .$$

From (7) and the semantic condition for inverse properties (OWL2/Tab5.12, "→") follows

$$(8)\ \forall x,\, y :\ \langle x,y\rangle \in \mathrm{IEXT}(p) \Leftrightarrow \langle y,x\rangle \in \mathrm{IEXT}(q)\ .$$

From (7) and the semantic condition for functional properties (OWL2/Tab5.13, "→") follows:

$$(9)\ \forall x,\, y_1,\, y_2 :\ \langle x,y_1\rangle \in \mathrm{IEXT}(q)\ \wedge\ \langle x,y_2\rangle \in \mathrm{IEXT}(q) \Rightarrow y_1 = y_2\ .$$

From (8) and (9) follows

$$(10)\ \forall y_1,\, y_2,\, x :\ \langle y_1,x\rangle \in \mathrm{IEXT}(p)\ \wedge\ \langle y_2,x\rangle \in \mathrm{IEXT}(p) \Rightarrow y_1 = y_2\ .$$

From (7) and the property extension of `owl:inverseOf` (OWL2/Tab.5.3) follows

$$(11)\ p \in \mathrm{IP}\ .$$

From (11), (10) and the semantic condition for inverse-functional properties (OWL2/Tab5.13, "←") follows

$$(12)\ p \in \mathrm{ICEXT}(I(\texttt{owl:InverseFunctionalProperty}))\ .$$

Since (12) follows from (5) and since p has been arbitrarily chosen, we receive

$$(13)\ \forall x : x \in \mathrm{ICEXT}(I(\texttt{ex:InversesOfFunctionalProperties})) \Rightarrow x \in \mathrm{ICEXT}(I(\texttt{owl:InverseFunctionalProperty}))\ .$$

From (1a) and the property extension of `owl:equivalentClass` (OWL2/Tab5.3) follows

$$(14)\ I(\texttt{ex:InversesOfFunctionalProperties}) \in \mathrm{IC}\ .$$

From OWL2/Tab5.3 follows for `owl:InverseFunctionalProperty`

$$(15)\ I(\texttt{owl:InverseFunctionalProperty}) \in \mathrm{IC}\ .$$

Finally, from (14), (15), (13) and the OWL 2 semantic condition for class subsumption (OWL2/Tab5.8, "←") follows

$$(16)\ \langle I(\texttt{ex:InversesOfFunctionalProperties}), I(\texttt{owl:InverseFunctionalProperty}\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ .$$

029_Ex_Falso_Quodlibet (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let x, y, l_1, l_2 be individuals such that

$$\begin{array}{l}
(1a1)\ \langle I(\texttt{ex:A}), I(\texttt{owl:Class})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
(1b1)\ \langle I(\texttt{ex:B}), I(\texttt{owl:Class})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
(1c1)\ \langle I(\texttt{ex:w}), x\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
(1d1)\ \langle x, l_1\rangle \in \mathrm{IEXT}(I(\texttt{owl:intersectionOf}))\ ,\\
(1e1)\ \langle l_1, I(\texttt{ex:A})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
(1e2)\ \langle l_1, l_2\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
(1e3)\ \langle l_2, y\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
(1e4)\ \langle l_2, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ ,\\
(1f1)\ \langle y, I(\texttt{ex:A})\rangle \in \mathrm{IEXT}(I(\texttt{owl:complementOf}))\ .
\end{array}$$

From (1d1), (1e1) – (1e4), and the semantic condition for class intersection (OWL2/Tab5.4, "→", binary) follows

$$(2)\ \forall z :\ z \in \mathrm{ICEXT}(x) \Leftrightarrow z \in \mathrm{ICEXT}(I(\texttt{ex:A}))\ \wedge\ z \in \mathrm{ICEXT}(y)\ .$$

From (1f1) and the semantic condition for class complement (OWL2/Tab5.4, "→") follows

$$(3)\ \forall z :\ z \in \mathrm{ICEXT}(y) \Leftrightarrow z \notin \mathrm{ICEXT}(I(\texttt{ex:A}))\ .$$

From (2) and (3) follows

$$(4)\ \forall z :\ z \in \mathrm{ICEXT}(x) \Leftrightarrow z \in \mathrm{ICEXT}(I(\texttt{ex:A}))\ \wedge\ z \notin \mathrm{ICEXT}(I(\texttt{ex:A}))\ .$$

From (1c1) and the RDFS semantic condition of ICEXT ("→") follows

$$(5)\ I(\texttt{ex:w}) \in \mathrm{ICEXT}(x)\ .$$

From (5) and (4) follows a contradiction, i.e. the set of premises is contradictory. From a contradiction follows arbitrary (*"ex falso sequitur quodlibet"*), hence we receive:

$$(6)\ \langle I(\texttt{ex:w}), I(\texttt{ex:B})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

030_Bad_Class (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be an x, such that

$$\begin{array}{l}
(1a)\ \langle I(\texttt{ex:c}), x\rangle \in \mathrm{IEXT}(I(\texttt{owl:complementOf}))\ ,\\
(1b)\ \langle x, I(\texttt{owl:Restriction})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ ,\\
(1c)\ \langle x, I(\texttt{rdf:type})\rangle \in \mathrm{IEXT}(I(\texttt{owl:onProperty}))\ ,\\
(1d)\ \langle x, I(\texttt{"true"8sd:boolean})\rangle \in \mathrm{IEXT}(I(\texttt{owl:hasSelf}))\ .
\end{array}$$

From (1a) and the semantic condition for class complement (OWL2/Tab5.4, "$\rightarrow$") follows

$$(2)\ \forall y:\ y \in \mathrm{ICEXT}(I(\texttt{ex:c})) \Leftrightarrow y \notin \mathrm{ICEXT}(x)\ .$$

From (1b), (1c), (1d) and the semantic condition for self-restrictions (OWL2/Tab5.6) follows

$$(3)\ \forall z:\ z \in \mathrm{ICEXT}(x) \Leftrightarrow \langle z, z\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type}))\ .$$

Now the following equivalence holds:

$$\begin{array}{ll}
\quad\langle I(\texttt{ex:c}), I(\texttt{ex:c})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:type})) & \\
\Leftrightarrow I(\texttt{ex:c}) \in \mathrm{ICEXT}(x) & :\text{by (3)}\\
\Leftrightarrow I(\texttt{ex:c}) \notin \mathrm{ICEXT}(I(\texttt{ex:c})) & :\text{by (2)}\\
\Leftrightarrow \langle I(\texttt{ex:c}), I(\texttt{ex:c})\rangle \notin \mathrm{ICEXT}(I(\texttt{rdf:type})) & :\text{by ICEXT definition}
\end{array}$$

Hence, we receive a contradiction solely from the original settings (1a), (1b), (1c) and (1d). That is, the original setting is an inconsistent ontology.

031_Large_Universe (Proof) Let I be an OWL 2 RDF-Based interpretation and B be a blank node mapping for the blank nodes in the premise graph such that $I + B$ satisfies the premise graph. Let there be x and l, such that the following holds:

$$\begin{array}{l}
(1a)\ \langle I(\texttt{owl:Thing}), x\rangle \in \mathrm{IEXT}(I(\texttt{owl:equivalentClass}))\ ,\\
(1b)\ \langle x, l\rangle \in \mathrm{IEXT}(I(\texttt{owl:oneOf}))\ ,\\
(1c)\ \langle l, I(\texttt{ex:w})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:first}))\ ,\\
(1d)\ \langle l, I(\texttt{rdf:nil})\rangle \in \mathrm{IEXT}(I(\texttt{rdf:rest}))\ .
\end{array}$$

From (1a) and the semantic condition for class equivalence (OWL2/Tab5.9, "$\rightarrow$") follows

$$(2)\ \forall z:\ z \in \mathrm{ICEXT}(I(\texttt{owl:Thing})) \Leftrightarrow z \in \mathrm{ICEXT}(x)\ .$$

From (1b), (1c), (1d) and semantic condition for enumeration classes (OWL2/Tab5.5, "$\rightarrow$") follows

$$(3)\ \forall z:\ z \in \mathrm{ICEXT}(x) \Leftrightarrow z = I(\texttt{ex:w})\ .$$

Since I is a simple interpretation and from the class extension of `owl:Thing` (OWL2/Tab5.2, "$\leftarrow$") we receive

$$(4a)\ I(\texttt{owl:Thing}) \in \mathrm{ICEXT}(I(\texttt{owl:Thing}))\ ,$$
$$(4b)\ I(\texttt{owl:Nothing}) \in \mathrm{ICEXT}(I(\texttt{owl:Thing}))\ .$$

Applying (2) and (3) to (4a) and (4b), respectively, leads to

$$(5a)\ I(\texttt{owl:Thing}) = \texttt{ex:w}\ ,$$
$$(5b)\ I(\texttt{owl:Nothing}) = \texttt{ex:w}\ ,$$

and therefore

$$(6)\ I(\texttt{owl:Thing}) = I(\texttt{owl:Nothing})\ .$$

From (6) and (4b) follows

$$(7)\ I(\texttt{owl:Nothing}) \in \mathrm{ICEXT}(I(\texttt{owl:Nothing}))\ .$$

However, from the class extension of `owl:Nothing` (OWL2/Tab5.2) follows

$$(8)\ \forall z:\ z \notin \mathrm{ICEXT}(I(\texttt{owl:Nothing}))\ .$$

By (7) and (8) we get a contradiction. Hence the original setting (1a), (1b), (1c) and (1d) is an inconsistent ontology.

032_Datatype_Relationships (Proof) Let I be an OWL 2 RDF-Based interpretation that satisfies the empty graph.

As a consequence of OWL2/Def4.2, I must be specified with respect to some OWL 2 RDF-Based datatype map D. According to OWL2/Def4.1, D must include the datatypes denoted by the URIs `xsd:string`, `xsd:integer`, and `xsd:decimal`. The denotations are given by name-datatype pairs "(u, d)" provided by the datatype map, and the value spaces are given as "VS(d)". According to the *"general semantic conditions for datatypes"* in the specification of D-entailment, the datatypes are identified by "I(`xsd:string`)", "I(`xsd:integer`)", and "I(`xsd:decimal`)", respectively. Secondly, the datatypes $I(u)$, for u one of "`xsd:string`", "`xsd:integer`", and "`xsd:decimal`", are instances of the set ICEXT(I(`rdfs:Datatype`)). OWL2/Tab5.2 implies ICEXT(I(`rdfs:Datatype`)) = IDC. From OWL2/Tab5.1 follows that IDC is a sub set of IC. From OWL2/Tab5.2 follows that ICEXT(I(`owl:Class`)) = IC. Hence, we get:

$$(1a)\ I(\texttt{xsd:string}) \in \mathrm{IC}\ ;$$
$$(1b)\ I(\texttt{xsd:integer}) \in \mathrm{IC}\ ;$$
$$(1c)\ I(\texttt{xsd:decimal}) \in \mathrm{IC}\ .$$

Further, according to the *"general semantic conditions for datatypes"* in the specification of D-entailment the datatypes have the following value spaces:

$\mathrm{ICEXT}(I(\texttt{xsd:string})), \mathrm{ICEXT}(I(\texttt{xsd:integer}))$, and $\mathrm{ICEXT}(I(\texttt{xsd:decimal}))$. According to OWL2/Def4.1 (referring to the OWL 2 Structural Specification), the value spaces of the three datatypes are defined according to the XSD Datatype specification. This has the following consequences. Firstly, the value spaces of `xsd:decimal` and `xsd:string` are disjoint sets:

$$(2a)\ \forall x:\ \neg[\, x \in \mathrm{ICEXT}(I(\texttt{xsd:decimal}))\ \wedge\ x \in \mathrm{ICEXT}(I(\texttt{xsd:string}))\,]\ .$$

Secondly, the value space of `xsd:integer` is a subset of the value space of `xsd:decimal`:

$$(2b)\ \forall x:\ x \in \mathrm{ICEXT}(I(\texttt{xsd:integer})) \Rightarrow x \in \mathrm{ICEXT}(I(\texttt{xsd:decimal}))\ .$$

Using (1c), (1a), (2a), and the "←" direction of the semantic condition for class disjointness (OWL2/Tab5.9), we get:

$$(3a)\ \langle I(\texttt{xsd:decimal}), I(\texttt{xsd:string})\rangle \in \mathrm{IEXT}(I(\texttt{owl:disjointWith}))\ .$$

Using (1b), (1c), (2b), and the "←" direction of the OWL 2 semantic condition of class subsumption (OWL2/Tab5.8), we get:

$$(3b)\ \langle I(\texttt{xsd:integer}), I(\texttt{xsd:decimal})\rangle \in \mathrm{IEXT}(I(\texttt{rdfs:subClassOf}))\ .$$

The combination of (3a) and (3b) was the conjecture.

C Translation into TPTP

In Section 3 it was explained how RDF graphs and the semantics of OWL 2 Full (see Section 2.1) are translated into FOL, and Section 4.1 mentioned that the *TPTP language* [14] is used as a concrete FOL serialization syntax. In this appendix, the translation into TPTP are demonstrated by means of a concrete example. The translation is demonstrated using the test case

020_Logical_Complications

from the test suite of *characteristic OWL 2 Full conclusions*, which has been defined in Appendix B. The example translation will be complete in the sense that the resulting TPTP encoding can be used with FOL ATPs that understand the TPTP language[7] in order to obtain the reasoning result of the test case. The TPTP translations for the example test case and for all other characteristic conclusions test cases are included in the electronic version of the test suite; see Appendix B for pointers. In addition, the *supplementary material* for this paper (see the download link at the beginning of Section 4) contains a translation of a large fragment of the OWL 2 Full semantics into TPTP (see Section 4.1 for a characterization of the fragment) and provides an executable software tool for the conversion of arbitrary RDF graphs into TPTP.

C.1 RDF Graphs and Test Case Data

In this section it is shown how RDF graphs and test case data are converted into the TPTP language. Example translations are given for the premise and conclusion graphs of the entailment test case *020_Logical_Complications* from the "characteristic OWL 2 Full conclusions" test suite.

According to Section B.1, the *premise graph* of the example test case is given in Turtle syntax[8] as:

```
@prefix ex:   <http://www.example.org/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

ex:c owl:unionOf ( ex:c1 ex:c2 ex:c3 ) .
ex:d owl:disjointWith ex:c1 .
ex:d rdfs:subClassOf [
    owl:intersectionOf (
        ex:c
        [ owl:complementOf ex:c2 ]
    )
] .
```

[7] The reasoners available online as part of the SystemOnTPTP service can be used for this purpose: `http://www.tptp.org/cgi-bin/SystemOnTPTP/`.

[8] Turtle RDF syntax: `http://www.w3.org/TeamSubmission/turtle/`

This encoding uses some of the "syntactic sugar" that Turtle offers for concisely representing certain language constructs, such as RDF collections. For the purpose of translating the RDF graph into TPTP, it is advisable to restate the above representation into an equivalent form that consists of only RDF triples:

```
@prefix ex:   <http://www.example.org/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

ex:c owl:unionOf _:lu1 .
_:lu1 rdf:first ex:c1 .
_:lu1 rdf:rest _:lu2 .
_:lu2 rdf:first ex:c2 .
_:lu2 rdf:rest _:lu3 .
_:lu3 rdf:first ex:c3 .
_:lu3 rdf:rest rdf:nil .
ex:d owl:disjointWith ex:c1 .
ex:d rdfs:subClassOf _:xs .
_:xs owl:intersectionOf _:li1 .
_:li1 rdf:first ex:c .
_:li1 rdf:rest _:li2 .
_:li2 rdf:first _:xc .
_:li2 rdf:rest rdf:nil .
_:xc owl:complementOf ex:c2 .
```

Premise graphs of entailment test cases are translated into TPTP *axiom* formulae. Following the explanation in Section 3 on how to translate RDF graphs into FOL, the translation into TPTP is as follows:

```
fof(testcase_premise, axiom, (
    ? [B_xs, B_xc, B_lu1, B_lu2, B_lu3, B_li1, B_li2] : (
          iext(uri_owl_unionOf, uri_ex_c, B_lu1)
        & iext(uri_rdf_first, B_lu1, uri_ex_c1)
        & iext(uri_rdf_rest, B_lu1, B_lu2)
        & iext(uri_rdf_first, B_lu2, uri_ex_c2)
        & iext(uri_rdf_rest, B_lu2, B_lu3)
        & iext(uri_rdf_first, B_lu3, uri_ex_c3)
        & iext(uri_rdf_rest, B_lu3, uri_rdf_nil)
        & iext(uri_owl_disjointWith, uri_ex_d, uri_ex_c1)
        & iext(uri_rdfs_subClassOf, uri_ex_d, B_xs)
        & iext(uri_owl_intersectionOf, B_xs, B_li1)
        & iext(uri_rdf_first, B_li1, uri_ex_c)
        & iext(uri_rdf_rest, B_li1, B_li2)
        & iext(uri_rdf_first, B_li2, B_xc)
        & iext(uri_rdf_rest, B_li2, uri_rdf_nil)
        & iext(uri_owl_complementOf, B_xc, uri_ex_c2) ))) .
```

The Turtle representation of the *conclusion graph* of the test case is given as:

```
@prefix ex:   <http://www.example.org/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix owl:  <http://www.w3.org/2002/07/owl#> .

ex:d rdfs:subClassOf ex:c3 .
```

Conclusion graphs of entailment test cases are translated into TPTP *conjecture* formulae, which is done as follows:

```
fof(testcase_conclusion, conjecture, (
    iext(uri_rdfs_subClassOf, uri_ex_d, uri_ex_c3) )) .
```

C.2 Semantic Conditions of the OWL 2 RDF-Based Semantics

In this section it is shown how the semantic conditions of the OWL 2 RDF-Based Semantics are translated into the TPTP language. An example translation is given for a small subset of semantic conditions that are sufficient to entail the conclusion graph of the entailment test case *020_Logical_Complications* from its premise graph. The selection of the small sufficient subset of semantic conditions was made based on the correctness proof for the test case, as given in Section B.2. The following semantic conditions are used to prove correctness:

– extension of property `owl:disjointWith` (Section 5.3 of OWL 2 RDF-Based Semantics);
– class complement (Section 5.4 of OWL 2 RDF-Based Semantics);
– binary class intersection (Section 5.4 of OWL 2 RDF-Based Semantics);
– ternary class union (Section 5.4 of OWL 2 RDF-Based Semantics);
– class subsumption, OWL version (Section 5.8 of OWL 2 RDF-Based Semantics);
– class disjointness (Section 5.9 of OWL 2 RDF-Based Semantics).

Semantic conditions are translated into TPTP *axiom* formulae, since, technically, they act as further premises in addition to the axiom that represents the premise graph of a test case. Following the explanation in Section 3 on how to translate semantic conditions into FOL, the translation into TPTP is as follows:

```
% extension of property owl:disjointWith
% (Section 5.3 of OWL 2 RDF-Based Semantics)
fof(owl_prop_disjointwith_ext, axiom, (
    ! [X, Y] : (
            iext(uri_owl_disjointWith, X, Y)
        => (
              ic(X)
            & ic(Y) )))) .
```

```
% class complement
% (Section 5.4 of OWL 2 RDF-Based Semantics)
fof(owl_bool_complementof_class, axiom, (
    ! [Z, C] : (
            iext(uri_owl_complementOf, Z, C)
        =>
            ( ic(Z)
            & ic(C)
            & ( ! [X] : (
                    icext(Z, X)
                <=>
                    ~ icext(C, X) )))))) .

% binary class intersection
% (Section 5.4 of OWL 2 RDF-Based Semantics)
fof(owl_bool_intersectionof_class_002, axiom, (
    ! [Z, S1, C1, S2, C2] : (
            ( iext(uri_rdf_first, S1, C1)
            & iext(uri_rdf_rest, S1, S2)
            & iext(uri_rdf_first, S2, C2)
            & iext(uri_rdf_rest, S2, uri_rdf_nil) )
        => (
                iext(uri_owl_intersectionOf, Z, S1)
            <=> (
                  ic(Z)
                & ic(C1)
                & ic(C2)
                & ( ! [X] : (
                        icext(Z, X)
                    <=> (
                          icext(C1, X)
                        & icext(C2, X) )))))))) .

% ternary class union
% (Section 5.4 of OWL 2 RDF-Based Semantics)
fof(owl_bool_unionof_class_003, axiom, (
    ! [Z, S1, C1, S2, C2] : (
           ( iext(uri_rdf_first, S1, C1)
            & iext(uri_rdf_rest, S1, S2)
            & iext(uri_rdf_first, S2, C2)
            & iext(uri_rdf_rest, S2, S3)
            & iext(uri_rdf_first, S3, C3)
            & iext(uri_rdf_rest, S3, uri_rdf_nil) )
        => (
```

```
            iext(uri_owl_unionOf, Z, S1)
        <=> (
              ic(Z)
            & ic(C1)
            & ic(C2)
            & ic(C3)
            & ( ! [X] : (
                    icext(Z, X)
                <=> (
                      icext(C1, X)
                    | icext(C2, X)
                    | icext(C3, X) )))))))) .

% class subsumption, OWL version
% (Section 5.8 of OWL 2 RDF-Based Semantics)
fof(owl_rdfsext_subclassof, axiom, (
    ! [C1, C2] : (
            iext(uri_rdfs_subClassOf, C1, C2)
        <=> (
              ic(C1)
            & ic(C2)
            & ( ! [X] : (
                    icext(C1, X)
                =>
                    icext(C2, X) )))))) .

% class disjointness
% (Section 5.9 of OWL 2 RDF-Based Semantics)
fof(owl_eqdis_disjointwith, axiom, (
    ! [C1, C2] : (
            iext(uri_owl_disjointWith, C1, C2)
        <=> (
              ic(C1)
            & ic(C2)
            & ( ! [X] : (
                ~ ( icext(C1, X)
                  & icext(C2, X) ))))))) .
```